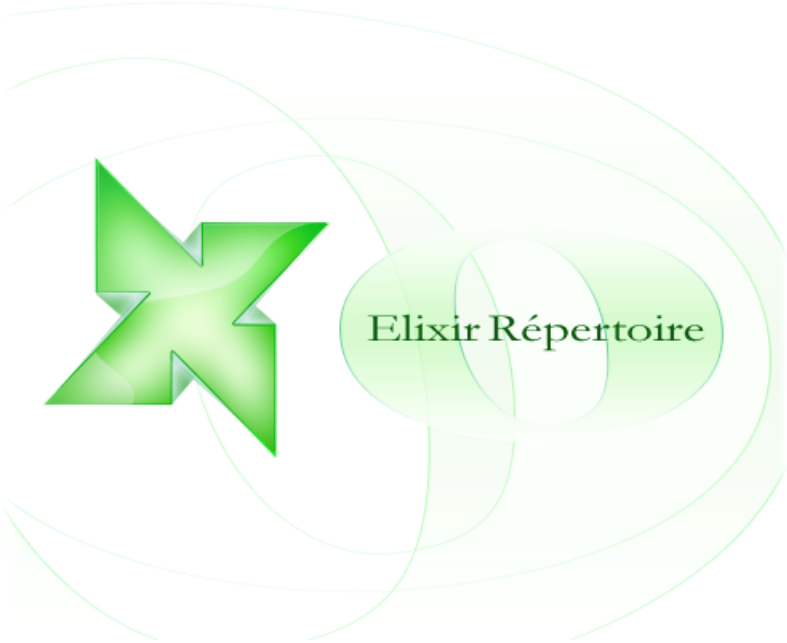
Elixir Report Designer User Manual

Release 8.0



Elixir **Technology Pte Ltd**

Elixir Report Designer User Manual: Release 8.0

Elixir Technology Pte Ltd

Published 2010
Copyright © 2004-2010 Elixir Technology Pte Ltd

All rights reserved.

Solaris, Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. Microsoft and Windows are trademarks of Microsoft Corporation.

Table of Contents

1. About Elixir Report Designer 1
    Enterprise Reporting for Web, Print and Mobile Delivery 1
    Look and Feel of Repertoire Designer 1
    Features of Report 1
    Report Security 2
    Working Set 3
    Hidden Files And Folders 4
2. Report Designer Workspace 5
    Adding a Report Template 5
        Blank Report 6
        Standard Report 7
    Case Study 10
3. Report Structure and Invocation 12
    Overview 12
    Elixir Report Designer Workspace 12
    Report Structure 13
        Report 13
        DataSources 14
        Page Setup 16
        Sections 17
        Styles 19
        Render Sequence 21
        Parameters 22
        Layout 22
4. Report Elements 26
    Overview 26
        Colours 26
        Common properties 28
    Types of Elements 31
        Manipulating Report Elements 32
        Selection 33
    Label 34
    Data Field 34
    Data Grid 37
    Image 40
    Check Box 41
    Line 42
    Rectangle 42
    Table 42
    Horizontal Box 44
    Vertical Box 44
    Page Break 45
    Sub-Report 45
        Sub Report example 46
    Barcode 48
        Barcode Types 48
    SVG 51
    Charts 52
        3D 54
        Area 54
        Bar 55
        Line 56
        Column 56
        Pie 56
        Meter 57

Polar 57
Stocks 57
Waterfall 57
XY 58
Chart Properties 58
Chart Preview on Workspace 59
Creating Charts 60
Creating Dual Axis Charts 63
Composite Chart 65
Combination Chart 66
Multiple Axes 66
RTF 66
Cube Table 67
Cube Table Properties 68
Header Properties 68
Measure Properties 68
Callback 69
Report Parts 70
Resizing report parts 70
Table of Contents (TOC) 70
Properties 71
Case Study 1 - Composite Chart 71
5. Report Rendering and Output Formats 73
Report Output formats 73
CSV 74
Glint 74
HTML 75
HTML-Zip 75
Image 76
IML 76
LPT 77
PCL 77
PDF 78
Print 83
PS 84
PPT 84
RTF 84
SVG-Zip 85
XLS 85
XML 86
Availability of Renderer 86
6. Scripting with JavaScript 87
Overview 87
Number 87
Boolean 87
String 88
Object 88
Array 88
References 89
Scriptlets 89
RenderIf 89
OnRenderBegin 89
OnRenderEnd 90
OnLayout 90
Script Editor 91
JavaScript Security 92
Introduction 92
Steps to protect users from malicious Javascripts 93

Configure Security Permissions 93
Verify that Security Policies are Taking Effect 94
JavaScript Cookbook 96
Alternating colours 96
Hiding and showing components 96
Using parameters to dynamically set values 97
Accessing Java classes 97
7. Elixir Report Designer JavaScript Reference 98
JavaScript API Reference 98
Elixir Utility Functions 98
Utility Objects 100
Data Objects 102
Raw Report Objects 104
Logical Report Objects 104
8. Office Report Template 106
DocX 106
Getting Started 106
Comments 106
DataSources 107
Parameter 107
Scripts 109
Aliases 109
Hide Processing Instructions 109
Formatting 110
Render DocX Report 110
9. Elixir Report Designer Migration Guide 112
Migration Guide 112
Migration Overview 112
Template Migration Steps 113
DataSource Migration Steps 113
Adding a DataSource 114
Batch Mode Template Migration 115
Changes and Improvements in Version 5 116
Data Sources 117
Report Elements 117
Special Parameters 117
Render Formats 117
Runtime API 117

List of Figures

1.1. Set Security Options 2
1.2. Global Properties - Repository 3
1.3. Working Set Wizard 4
2.1. Add Report 5
2.2. Choose DataSource 6
2.3. Choose Report Type 6
2.4. Standard Reports 7
2.5. Generated Tabular Report 7
2.6. Generated Columnar Report 8
2.7. Select Fields 9
2.8. Label Settings 10
2.9. Generated Glint Report 11
3.1. Report Layout 13
3.2. Report Tree 13
3.3. Report Properties 14
3.4. Add Filter For DataSource 15
3.5. Page Setup 16
3.6. Section Wizard 17
3.7. Difference between Sort and Sort (Simple) 19
3.8. Style Wizard 20
3.9. Render Sequence 21
3.10. Render Wizard 23
3.11. Sample Layout 24
4.1. Choose Colour 26
4.2. Named Colours 27
4.3. Choose Colour (Excel Compatible) 28
4.4. Borders and Backgrounds 28
4.5. Sample Data 35
4.6. Sample Running Sum Report 36
4.7. Running Sum Output 36
4.8. Table Wizard 43
4.9. Select Fields for Table 44
4.10. SubReport Example Layout 47
4.11. SubReport Output 48
4.12. Barcode Wizard 50
4.13. Barcode Control Source 50
4.14. Barcode Options 51
4.15. Positive and Negative Values Positioning with Interval Marker from 10 to 25 53
4.16. Chart Wizard 53
4.17. XYZ Axes 54
4.18. Bar Chart without Sub-Category 55
4.19. Bar Chart with Sub-Category 56
4.20. Sample Waterfall Chart 58
4.21. Properties Panel 60
4.22. Chart Preview Disabled 60
4.23. Column Chart 62
4.24. Inverted Data Column Chart 62
4.25. Dual Axis Chart 63
4.26. Dual Axis Chart Design 64
4.27. Composite Chart Type 65
4.28. Add Subplot 66
4.29. Sample Cube Table 67
4.30. Cube Table Designer View 67
4.31. Shape Tree View 68
4.32. Measure Presentation Options 69

4.33. Composite Chart 72
5.1. Render Wizard 73
5.2. CSV Options 74
5.3. IML Options 77
5.4. PCL Wizard 78
5.5. PDF Options 79
5.6. PDF CJK Mapping 80
5.7. Print Options 83
6.1. Script Editor 92
6.2. Generated Report Before Changes 94
6.3. Additional Scripts 95
6.4. Generated Report After Changes 95
6.5. Error logs in Console 96
8.1. Parameters in DocX Report 108
8.2. Rendered DocX Report 108
8.3. Render DocX 111
9.1. Import Wizard 113
9.2. Report Elements 114
9.3. Configure DataSource Wizard 115

List of Tables

4.1. URL Targets 31
5.1. Java AWT font types on Window platform. 81
5.2. Type1 PDF fonts with encoding Cp1252 82
5.3. List of PDF Type 1 font names 82
5.4. List of possible encodings 82
6.1. JavaScript Keywords 87
7.1. Core Object, Number and String functions 99
7.2. Date functions 99
7.3. Format 100
7.4. Log 100
7.5. Properties 101
7.6. Renderer 101
7.7. Data 102
7.8. DataCache 103
7.9. DataCacheManager 103
7.10. Function 104
7.11. GroupNode 104
8.1. Comments 106
8.2. Formatting Syntax 110

Chapter 1
About Elixir Report Designer

Enterprise Reporting for Web, Print and Mobile Delivery

Amidst fast-changing business environment, corporate enterprises need timely and accurate visibility into business operations. Actionable information must be presented and delivered to the right targets in the right forms. Designed for power and ease of use, Elixir Report Designer brings unmatched price-performance to meet your enterprise wide reporting requirements.

Sporting a new-generation End-to-End Integrated Business Intelligence Architecture, Elixir Report Designer comes built-in with a data aggregation and transformation engine to enable complex data processing including OLAP Cube transformation. With newly enhanced advanced internationalization support, Elixir Report Designer pushes the envelope for multi-lingual support to enable enterprises tapping into the increasingly globalized markets. Numerous innovative report layout controls deliver unsurpassed power to report developers, while keeping ease of use and maintenance in balance. Complex layout requirement is no more a problem with advanced features for templates spanning across multiple pages, or with multiple report sections with respective header and footer control, auto-resizable horizontal and vertical boxes, user-definable style support for CSS Level 3 - complete with the full range of exotic border styles such as Groove, Inset, and Wave, as well as an all-encompassing dynamic watermark control which can include just about anything a report can render.

As an integral part of Elixir's integrated Business Intelligence suite covering the full information lifecycle from ETL, OLAP, to Dashboard, Elixir Report Designer is the new choice for Enterprise Reporting for Web, Print and Mobile Delivery, a fundamental building block for enabling the Information Enterprise.

Look and Feel of Repertoire Designer

The look and feel of Repertoire Designer can be edited by user by changing it in `Global Properties` then `Look and Feel` tab. In order to have the `Look and Feel` tab appear in `Global Properties`, *substance-lite.jar* and *substance-theme-pack.jar* must be placed in /Repertoire/ext. Only then, the user will be able to change the look and feel of Repertoire Designer using the list of options available.

Features of Report

Multiple DataSources: Multiple DataSources can be used in the same report or can be shared among various reports. The various data source types are JDBC, JDBC (JNDI), LDAP, XML, Text, Java Object. Through ODBC you can connect to a wide variety of databases such as Oracle, IBM, DB2, etc.

Rich Set of Report Elements: Standard report elements include barcode, data field, images, labels, etc. and Complex report elements include SVG, RTF, HTML and table components.

Scripting and Parameterization: With a powerful JavaScript engine, pre-built functions like time and date formatting, mathematical calculation, and other common operations. Reports can be parameterized to generate dynamically based on values passed in at runtime.

Flexible Report Formatting: Create reports of virtually any type ranging from contact listing and directory, sales reports, image-rich layouts and confidential documents with water marks.

Multiple Report Output Formats: The reports can be generated into various output formats like PDF (with Access Security), HTML, XML, CSV, etc.

Internet: Elixir Report now supports the ability to view your report in the web page using an applet and a Java plug-in through a web server.

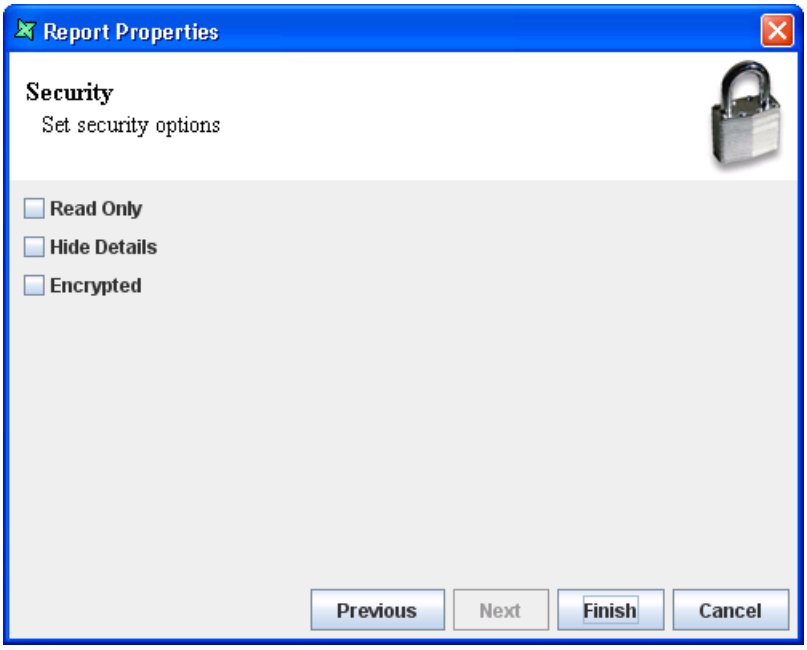
SVG: Report provides support for HTML and SVG components for producing high- resolution image output.

Mobile Support: Elixir Report Designer also enables integrated information delivery and reporting of Java and XML applications to users of mobile devices, in particular users of wireless Java devices. This is known as Elixir Report Mobile Edition. The ERME Designer component extends the Elixir Report Designer to allow creation of report templates for reports on mobile devices.

Report Security

At the last page of the Report Properties Wizard, the user can set security options, as seen in Figure 1.1, “Set Security Options”.

Figure 1.1. Set Security Options



- *Read-Only*: When selected and saved, the next time a user opens this report, the user will not be able to edit any details of the report template like name and scripts of the report template.
- *Hide Details*: When this option is selected and saved, the next time when this report is opened, the user will only be able to see the name of the report.
- *Encrypted*: This option is to be used with either `Read-Only` option or `Hide Details` option or both. Checking the checkbox will prompt the user to enter a password, then re-enter to confirm

the password. (Both passwords must be the same) After this is done, if another user would like to edit any selections, he will need to enter the password.

Working Set

Users can define a Working Set whereby only the selected filesystems will be displayed in *Elixir Repository*.

This is done by clicking on the `Global Properties` button on the `Action Bar`. Select the `Repository` tab, as shown in Figure 1.2, “Global Properties - Repository”. Click on the `Add` button to add a working set. The Working Set Wizard will appear as shown in Figure 1.3, “Working Set Wizard” and allow the user to enter the name of working set and to select the filesystems to add to the working set. After that, click on `Ok`, then select the desired working set from the drop-down list.

Figure 1.2. Global Properties - Repository

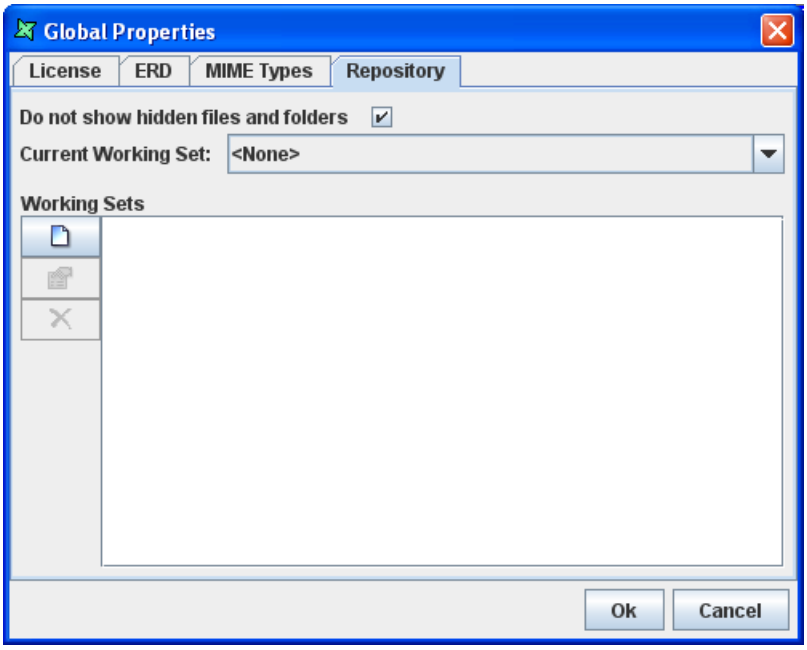
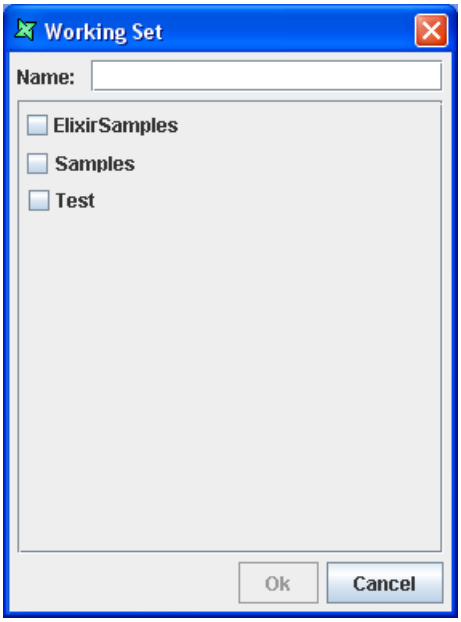


Figure 1.3. Working Set Wizard



Hidden Files And Folders

To set files or folders to hidden in a local filesystem, user will need to go to the directory where the file or folder is placed through Windows Explorer. Right-click on the file or folder and select `Properties`. Check the property that says `Hidden`. Click on `Apply` then `Ok`.

Refresh the filesystem in the repository. The file or folder that has the hidden property applied will no longer appear in the repository.

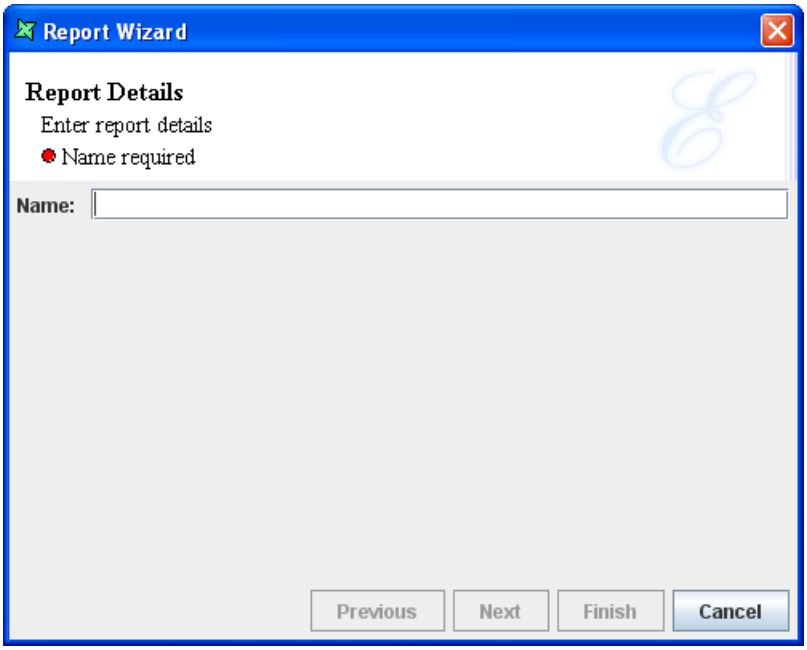
Chapter 2
Report Designer Workspace

Adding a Report Template

The common elements in the Elixir Workspace are discussed in the Elixir Repertoire user manual, which shows how to add filesystems and files. This chapter will discuss the Elixir Report Designer features that build on the workspace.

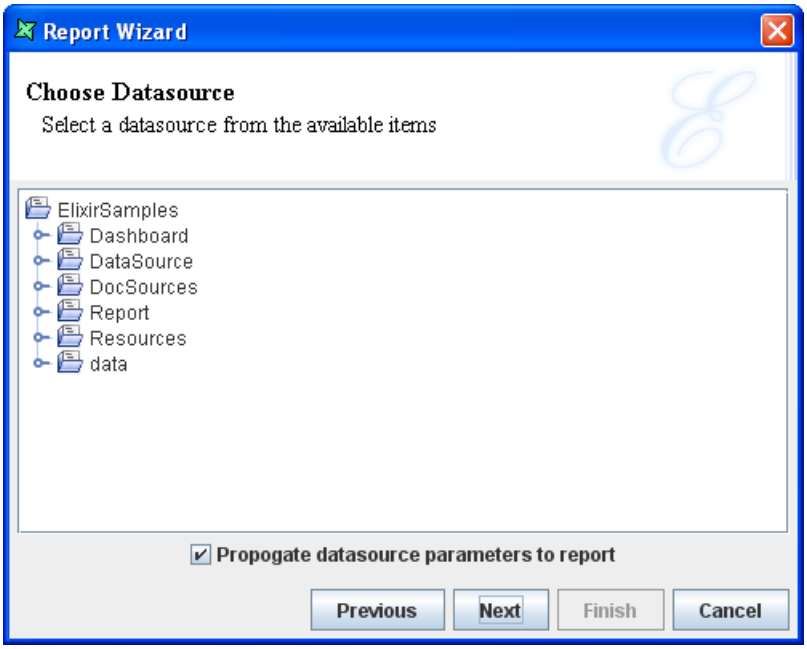
Each FileSystem has a pop up menu. On selecting the Add -> Report Template, the "Report Wizard" appears as shown in Figure 2.1, “Add Report”. Enter the name of the Report Template in the text box and click the Next button. Alternatively, by clicking the New File button [icon] on top of the Elixir Repository panel you can select Report Template from the Add File dialog. This is followed by selecting the location for the Report Template, then click on the Finish button.

Figure 2.1. Add Report



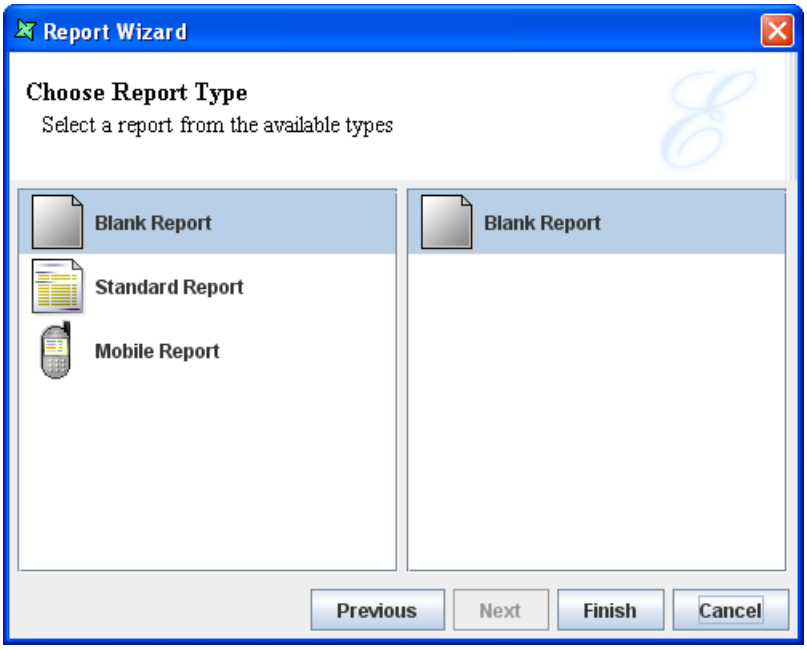
Choose a DataSource from the repository as shown in Figure 2.2, “Choose DataSource”. If the selected data source contains parameters that need to be propogated to the report then select "Propogate datasource parameters to report" check box. Otherwise you should provide values for any dynamic parameters here. For more information on the parameters and their propogation refer to Chapter 3, *Report Structure and Invocation*.

Figure 2.2. Choose DataSource



On the next screen, select the Report Type from the list of report types as shown in Figure 2.3, “Choose Report Type”.

Figure 2.3. Choose Report Type



There are wizard pages to guide you in creating three types of reports: Blank, Standard and Mobile.

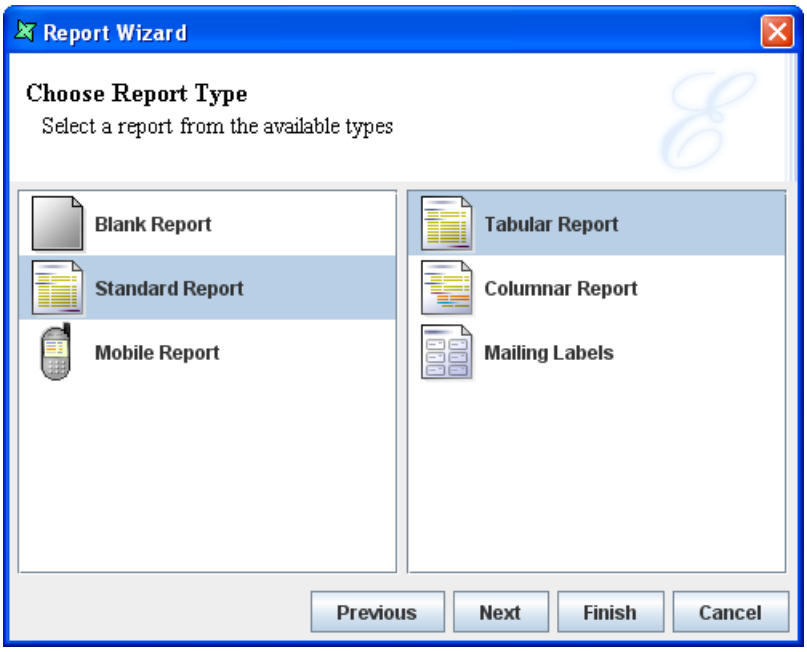
Blank Report

On selecting the Blank Report and clicking the Finish button a blank report is added to the window. All report components must be added manually.

Standard Report

The three types of Standard Reports are Tabular, Columnar Report and Mailing Labels as shown in Figure 2.4, “Standard Reports”.

Figure 2.4. Standard Reports



Tablular Report

If you want to add a report in the form of table then select the Tabular Report from the standard report types that are listed. Click the Next button.

The fields in the data source are listed so that you can choose the ones you want to add to the report. Here's an example of a tabular report generated with four columns:Figure 2.5, “Generated Tabular Report”.

Figure 2.5. Generated Tabular Report

SampleTable

first_name	last_name	position_title	salary
Sheri	Nowmer	President	80000.0
Derrick	Whelply	VP Country Manager	40000.0
Michael	Spence	VP Country Manager	40000.0
Maya	Gutierrez	VP Country Manager	35000.0
Roberta	Damstra	VP Information Systems	25000.0

You can now improve the report by adding colours, logos and additional information to meet your requirements.

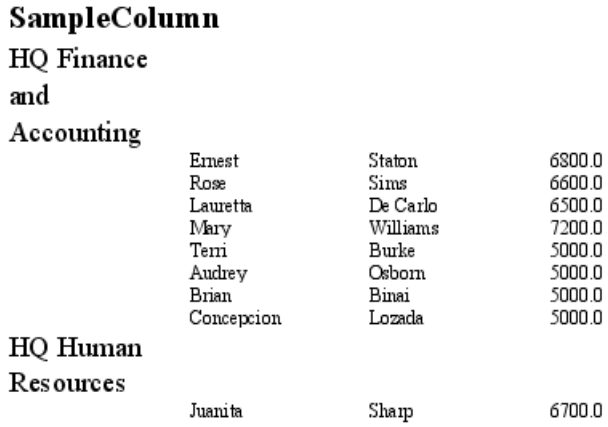
Columnar Report

A Columnar Report allows elements to be grouped. The creation steps are similar to the creation of a Tabular Report, except in addition you need to specify the fields to be grouped, and optionally any sorting. You are provided with some choices for the report layout:

- *Stepped*: Stepped layout is the default and the most commonly required layout for the majority of reports. In stepped layout the grouped fields are arranged in indented steps and the rest of the fields are arranged in sequence.
- *Block*: A block report displays data that has been grouped or sorted in vertical columns.
- *Align Left*: In this type of layout all the data are aligned to the left of the table. The grouped columns are displayed one below the other and the sorted columns are arranged vertically as in a table.
- *Outline*: It is similar to the stepped report except that there is outlining and highlighting of the group sections.

A sample stepped columnar report is shown in Figure 2.6, “Generated Columnar Report”.

Figure 2.6. Generated Columnar Report

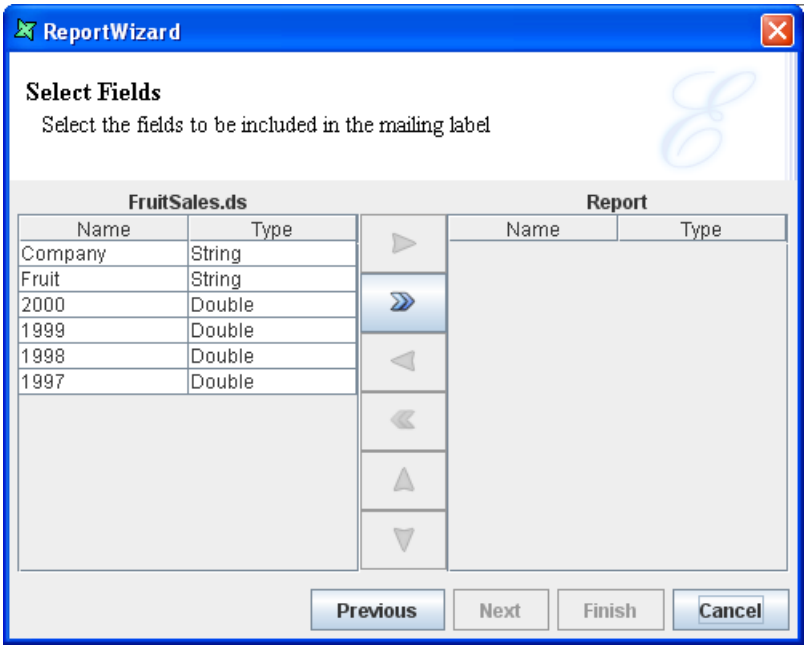


You can now improve the report by adding colours, logos and additional information to meet your requirements.

Mailing Labels

If you want to add a report in the form of a Mailing Labels then select "Mailing Labels" from the list of Report types. Click the Next button. The fields in the data source are listed as shown in Figure 2.7, “Select Fields”. Select the field that has to be included in the report and click on the '>' button. If all the fields have to appear in the report then click the '>>' button. After selecting the fields to be added in the report, click the Next button.

Figure 2.7. Select Fields



The screen appears with three tabbed panes. They are the *Label*, *Page* and *Preview*. Checking on `Custom Label` will enable the user to customize the Mailing Label based on individual's needs. If not, the fields in the respective tabs will not be editable.

The label settings are displayed in the Label tab shown in Figure 2.8, “Label Settings”. The available options under `Label` tab is given below:

- Label Width: Width of label
- Label Height: Height of label
- Horizontal Gap: Amount of space between each label horizontally
- Vertical Gap: Amount of space between each label vertically
- Left Padding: Amount of space on the left of Mailing Label from the page margin
- Top Padding: Amount of space at the top of Mailing Label from the page margin

Figure 2.8. Label Settings

Setting the size of the page and page margin are done in the `Page` tab. If the list of *Paper Size* is not enough to satisfy, user can specify their own dimensions to customize their template.

User can make use of the `Preview` tab to preview the template before creating the template. To complete the template creation process, click on the `Finish` button.

Case Study

Before creating a Report Template you should initially add a data source say, customer.ds to the FileSystem. Refer to the JDBC chapter of the Elixir Data Designer Documentation for details on how to add the JDBC data source Customer.ds.

If you want to add a simple tabular report follow the steps given below:

1. Select a FileSystem. Select Add->Report template from the file system popup menu.

2. When the Report Template Wizard appears, enter a name for the Report, `First report`, and click the Next button.

3. In the screen that appears locate and select the Customer.ds DataSource and click the Next button.

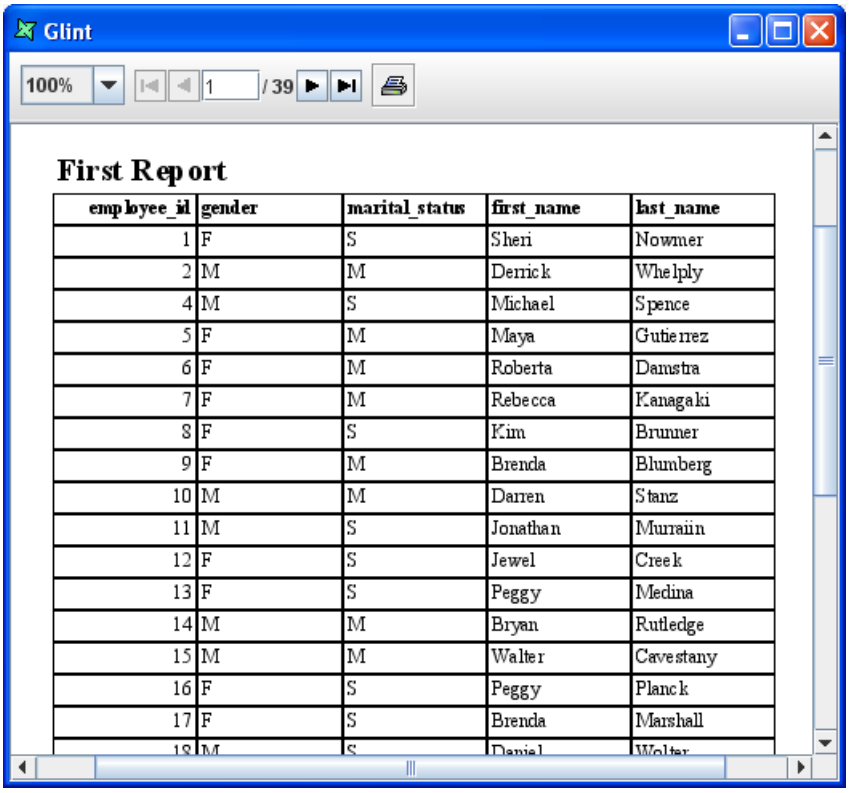
4. Select Standard Report. The different types of Standard report are listed on the right of the dialog. Select Tabular Report from the list and click on the Next button.

5. The next page allows you to choose the fields to show in the table. In this case, click the '>>' button to move all the fields to the Report table.

6. Finally, on clicking the Finish button, First report.rml is added to the repository and opened to show the Layout tab.

7. Select Render from the toolbar. On the Render Wizard select Glint and click the Finish button. The output is displayed as shown in Figure 2.9, “Generated Glint Report”.

Figure 2.9. Generated Glint Report



First Report

employee_id	gender	marital_status	first_name	last_name
1	F	S	Sheri	Nowmer
2	M	M	Derrick	Whelply
4	M	S	Michael	Spence
5	F	M	Maya	Gutierrez
6	F	M	Roberta	Damstra
7	F	M	Rebecca	Kanagaki
8	F	S	Kim	Brunner
9	F	M	Brenda	Blumberg
10	M	M	Darren	Stanz
11	M	S	Jonathan	Murraiin
12	F	S	Jewel	Creek
13	F	S	Peggy	Medina
14	M	M	Bryan	Rutledge
15	M	M	Walter	Cavestany
16	F	S	Peggy	Planck
17	F	S	Brenda	Marshall
18	M	S	Daniel	Wolter

Chapter 3
Report Structure and Invocation

Overview

In this chapter you will learn about the basic structure of reports and the different sections within it. You will also get to know about the section invocation and how to control the render sequence.

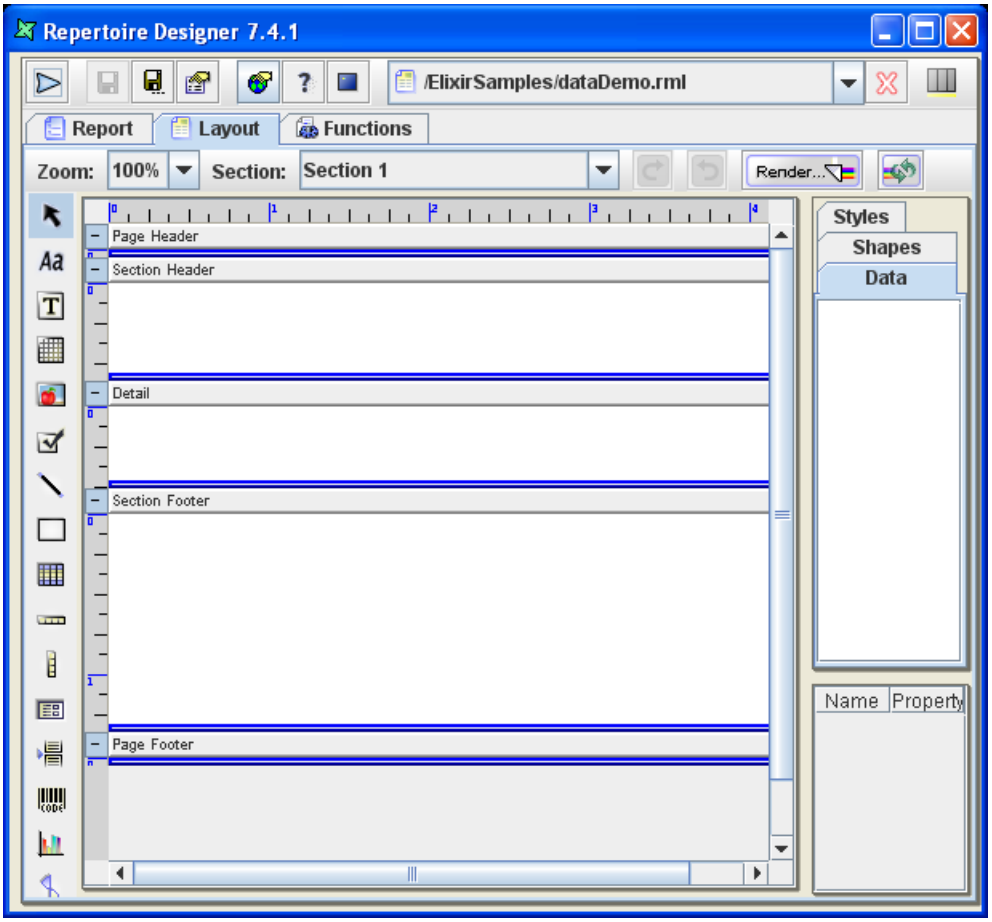
Elixir Report Designer Workspace

The Designer workspace consists of three tabbed panes. They are Report tab, Layout tab and the Functions tab.

The report structure, consisting of the DataSources, Page Setup, Sections and Style sections are displayed in the left panel of the Report tab window. The Render Sequence and the Parameters panel are present on the right of the Report tab window. These will be discussed later in this chapter.

The Layout tab window contains a tool bar and report layout as shown in Figure 3.1, “Report Layout”. The layout contains a detail, headers and footers.

Figure 3.1. Report Layout

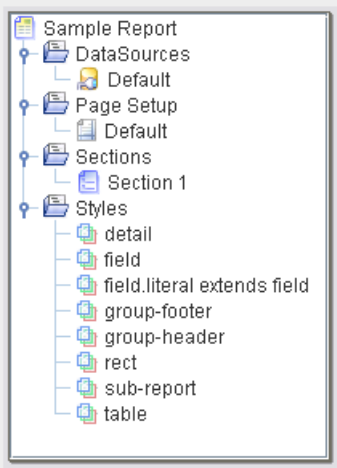


The Functions tab window contains several tabbed panes: Script Summary, Function Definitions, On Render Begin, On Render End and Render If. These will be described in Chapter 6, *Scripting with JavaScript*.

Report Structure

In this section we will discuss the Report and Layout panels.

Figure 3.2. Report Tree



The Report tab includes a tree view of the structure of the report template, the Render Sequence panel and the Parameter panel. The structure of a report template is shown in Figure 3.2, “Report Tree”. Each template consists of the DataSources, Page Setup, Sections and Styles.

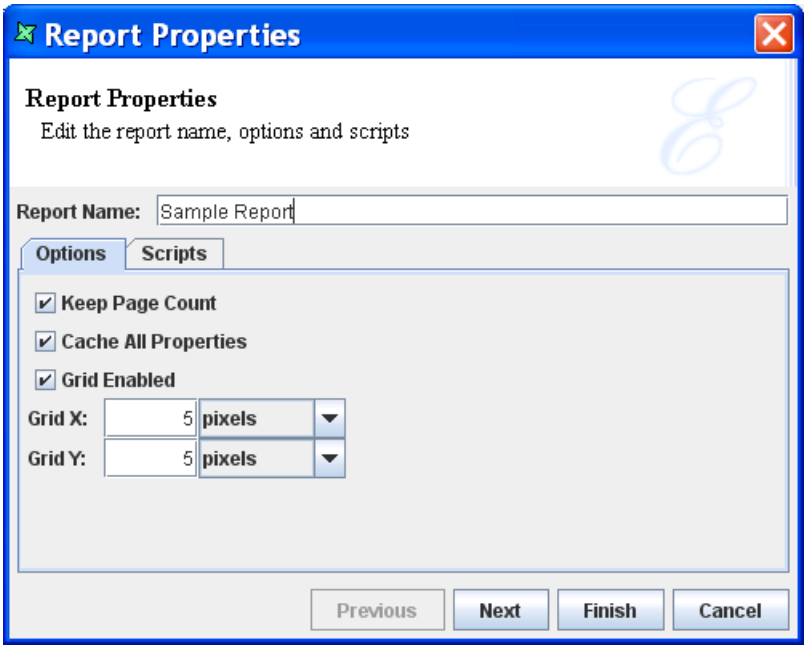
Report

The root item of the tree is the report itself. From the popup menu you have access to the Report Properties (Figure 3.3, “Report Properties”. Here you can change the report name, control settings that apply across the report and control any RenderBegin and RenderEnd scripts for the report as a whole. The RenderBegin script will run before the first section is rendered. This means that it happens before the render sequence is used and before any data is loaded. Therefore you can control the render sequence programat-

ically from here. The RenderEnd script is run after all appropriate sections have been completely rendered, allowing some clean up, if necessary.

Keep Page Count this property is enabled by default, allowing you to use page count substitutions (ie. ${##}) within your report. Page counting is expensive because the whole report needs to be kept in memory until the page count is known, only then can the first page be output (presumably showing the page count). If you don't need to show an overall page count you will get much better memory use by turning this off. PDF output format is special, in that it allows page counts to be forward referenced, so page counts will still work in PDF even with this option disabled. Therefore for optimal PDF output (smallest memory use), you should disable this. In all other formats, you will get a value of 0 for page count if this flag is not enabled.

Figure 3.3. Report Properties



Cache All Properties this property is enabled by default. When rendering a report, each component has an explicit or implicit style associated with it. Searching for the appropriate property to apply requires checking both the parent hierarchy and any style hierarchy. This is expensive to perform repeatedly. If you know that the styles will not change while the report is being rendered, it is best to cache the styles at the start. This means any subsequent scripted style changes won't be used. If you are manipulating styles at runtime, using scripts, then set this value to false, to ensure the renderer knows the styles are volatile and always identifies the correct style property.

Grid Enabled this property enables users to define a snap-to-grid for all components in the report template. This allows for accurate placement and sizing of report components.

Note

Once you have positioned your components, you can use the Lock property available for each component on the property sheet. This will prevent the components being accidentally nudged.

DataSources

If a datasource was chosen from the Add Report Template wizard then a reference to it will be included as the 'Default' DataSource. A datasource reference is a short name for a datasource file in the repository. For example, the full path might be "repository:/Workspace/Samples/Employee.ds", but the reference name can just be "Employee". A datasource reference also holds values for any dynamic parameters

that the datasource requires. You can add or modify the datasources that the report references using the popup menus in the tree. A report can only use datasources that are listed here, and refers to them by their reference name. Any number of data sources can be defined; each must have a reference name. You can include the same datasource more than once (with different reference names), to allow variations in dynamic parameters, or independent iteration through the records.

Each datasource may be accessed in one of two modes: Table mode or Streamed mode. There is a checkbox on the DataSource wizard to control this. The default is Table mode. When operating in Table mode, the records are loaded into memory so that they can be sorted and grouped and so that you can query the total number of records etc. In Streamed mode, only a single record is loaded into memory at one time, which greatly improves the memory efficiency, but prevents complex operations that require access to more than one record. Streamed mode is best suited for billing operations etc. where each customer record has no connection with prior or subsequent records. If you turn off Table mode (therefore adopting Streamed mode) then any operations which depend on the presence of the table, such as sorting, grouping and cross-record operations will fail.

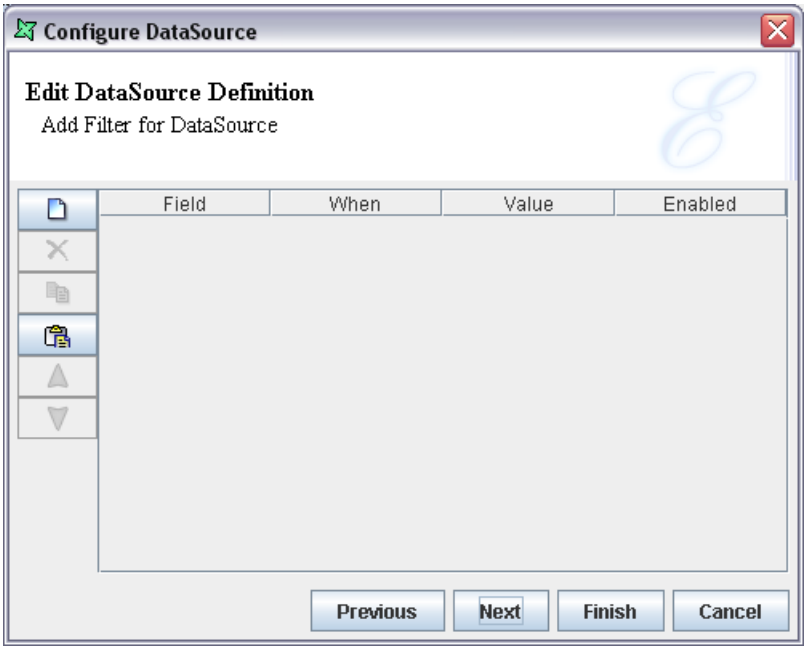
There are two ways of opening a datasource when a report template is already opened. First, in the Data|Shapes|Styles tree panel, right-click on the datasource then select `Open DataSource`. The same procedure applies to the datasource(s) listed under the Report tab.

A DataSource reference can be copied using the popup menu and pasted into another report template. It can also be pasted into the same report, to make a copy before modifying it.

From Repertoire 7.4.0 onwards, datasource(s) added in the *Report Tree* can do filtering of data without the use of Composite datasource. However, this is only restricted to filtering. In order to do other processing, Composite datasource must be used. To filter, right-click or double-click on the datasource listed in the *Report Tree* then click `Next`, which leads to the page to add filter(s) to the datasource, as seen in Figure 3.4, “Add Filter For DataSource”.

The other way of doing it will be through the `Data` tab on right of report layout. Right-click on the datasource and select `Edit DataSource...`, then the `Next` button. The page will be identical to the one seen in Figure 3.4, “Add Filter For DataSource”.

Figure 3.4. Add Filter For DataSource



Page Setup

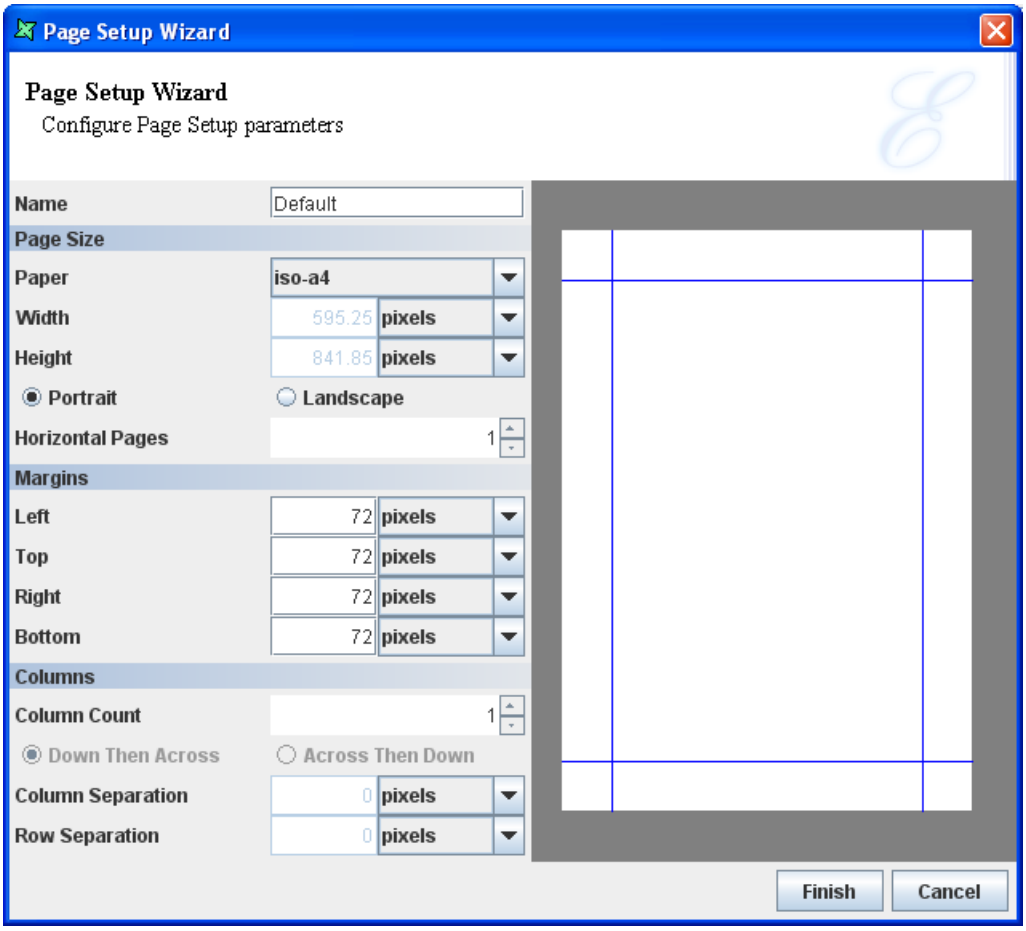
When a Report Template is created, a 'Default' Page Setup will be added under the Page Setup section of the tree. Each section of the report can have it's own page setup, or several can share the same one. This allows reports to combine landscape and portrait sections, or multi-column and single column sections within the same report. Popup menus on the tree provide the usual abilities to add or modify page setups. The Page Setup Wizard appears as shown in Figure 3.5, “Page Setup”.

The paper type is selected from the Paper combo box. The width and height of the paper type will be automatically entered in the corresponding text boxes and the unit of measurement from the combo boxes.

The page orientation, either Portrait or Landscape, can be selected by clicking on the corresponding option button. The number of horizontal pages that the report will span have to be specified in the Horizontal Pages field. The left, right, top and bottom margins are entered in the corresponding text boxes and the unit of measurement from the combo box. You can set the unit of measurement you want to use by default from the Global Properties button on the workspace toolbar.

The Column count is entered in the text box. Most reports are "Down Then Across", which means the content flows down one column before proceeding to the next. Some templates, such as labels may be need to be "Across Then Down" - each band of the report, eg. each detail is laid out in subsequent columns of the same row, returning to the first column of the next row when a row is complete.

Figure 3.5. Page Setup



The Column Separation specifies the space between two columns. Similarly, the Row Separation corresponds to the space between the rows of data.

A Page Setup can be copied using the popup menu and pasted into another report template. It can also be pasted into the same report, to make a copy before modifying it.

Note

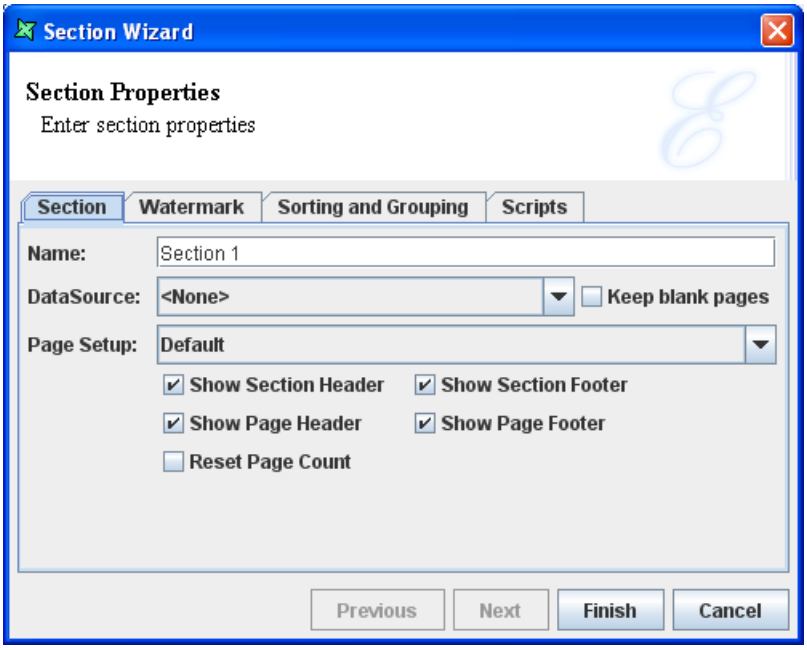
By default the measurement units of all the fields in the Page Setup is set to pixels. If desired you can change the default measurement type. This can be done by clicking on the Global Properties icon. The Global Properties Wizard appears. In the ERD tab, select the measurement type from the Default Units combo box and click on the Ok button.

Sections

Every report template will have at least one section. A new section can be added using the popup menu of the Sections node. The newly added section will be added to `Render Sequence` automatically.

The Sections Properties Wizard appears as shown in Figure 3.6, “Section Wizard”. It consists of four tabs: Section, Sorting and Grouping, Watermarks and Scripts.

Figure 3.6. Section Wizard



Section

The name of the section is entered in the text box and the datasource is selected from the DataSource combo box. A section can have no datasource, in which case it will still have a section header and footer, but no details will be generated. This can be useful for special purpose sections, such as cover pages or watermarks.

The "Keep blank pages" check box is selected if any blank pages produced during generation should be preserved. Blank pages are usually the result of combinations of page breaks due to the way the report is defined and should usually be discarded. Each section may contain a page header, section header, section footer and page footer. A section header is shown before any details, starting on the first page and a section footer is shown following any details.

By default each section after the first one will increment the page number as a continuation of the previous section. If you need the page number to restart again at 1 then select the "Reset Page Count" checkbox. This will break the report into logical page sequences which will each have individual

numbering and page totals. You can also begin a new page sequence within a section by adding an explicit page break with a Reset Page Count option.

A Section can be copied using the popup menu and pasted into another report template. It can also be pasted into the same report, to make a copy before modifying it. A report can have any number of sections.

Within a section, you can refer to fields belonging to the section datasource by their field names. You can also refer to fields belonging to other datasources, but you need to refer to them by their full name, for example the field Salary in the datasource named Employee would be known as Salary if Employee was the section datasource or as Employee:Salary otherwise. You can only refer to fields in datasources that are referenced by the report template.

Watermark

A watermark can be a valuable addition to your report – whether you want to enhance the appearance of the document by adding a seal or image or whether you want to add a text watermark that identifies the report contents as a draft or confidential information, the watermark feature in report is a definite asset for the users.

A watermark may contain graphics and text that appears behind the main text of the report. It is usually a lighter shade than the text, so you can read the report easily. Watermarks add visual attention and a professional look with the use of imagery such as logos.

Watermarks are very flexible in Elixir Report Designer - a watermark is just another section, either belonging to this template or a section from any other template in the repository, that is shown behind the main information on page. This means anything you can do in a section can be done in a watermark. A watermark can contain traditional text and images, but also data, charts, tables subreports etc.

When using a section as a watermark you need to consider it's data requirements. If the section has no datasource, for example it is just an image or literal text, you can leave the datasource blank. Also, if the section has a datasource then that source will be used by default, there's no need to redefine it here. The only time you need to set the watermark datasource is if you need to supply a missing datasource (the section needs one, but doesn't define one) or if you want to override the existing datasource.

Sorting and Grouping

Grouping is done for categorization of data into various logical collections. Sorting is to arrange the data in ascending/descending order.

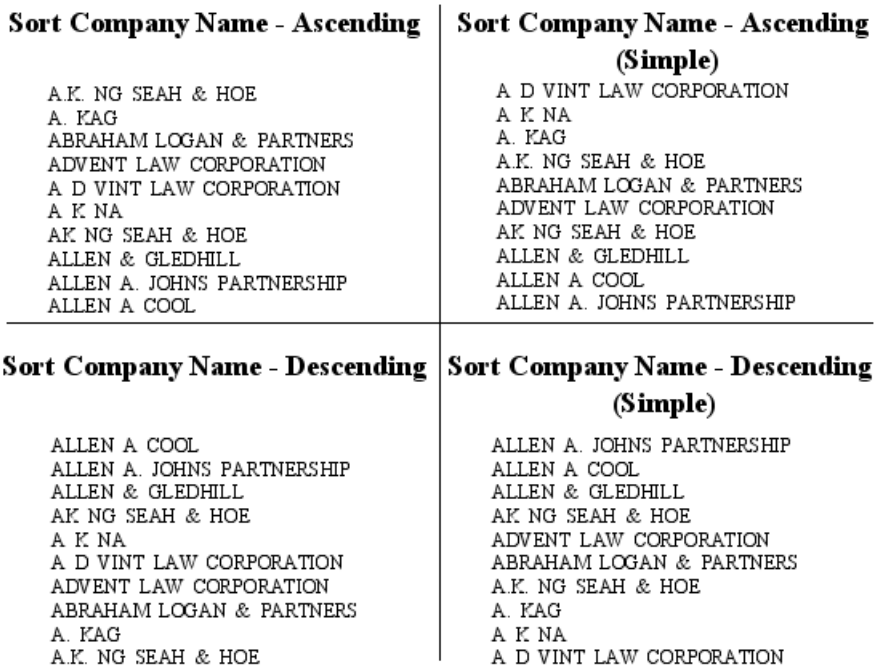
In the Sorting and Grouping tab there are four columns Name, Sort Order, Group On and Group Data and 5 buttons Add, Edit, Move Up, Move Down and Remove.

On clicking the Add button the Sort dialog appears. Select the name of the Field to be sorted from the combo box. Select Ascending, Descending, Ascending(Simple) or Descending(Simple) Sort Order from the combo box.

The difference between Sort (Ascending, Descending) and Sort(Simple) Order:

- Sort follows a proper dictionary sort algorithm whereby spaces are ignored. Sort(Simple) doesn't follow the proper full sort definition, instead it treats every character as significant.
- Dictionary sort in the order of symbols first followed by alphabet, ignoring any spaces in-between the words. Sort(simple) is in the order of spaces, symbols then alphabet.
- Sort(Simple) is especially useful when sorting data that contain any symbols, abbreviation, initials and spaces. An example as shown in Figure 3.7, “Difference between Sort and Sort (Simple)”.

Figure 3.7. Difference between Sort and Sort (Simple)



If your datasource is already sorted (eg. as a result of an SQL "ORDER BY" clause), then you can select None here. Select the Group On option from the combo box. If a range, substring, etc. is selected as a Group On option then a text box appears where we can specify the value for that range or substring based on which the grouping will be done. The Show Group Header and Show Group Footer check boxes are selected if required. On clicking the Ok button after entering the values in the Sort dialog they are assigned to the corresponding Name, Sort Order, Group On and Group Data columns in the Wizard.

Scripts

There are three text fields for Render If, On Render Begin and On Render End functions. The JavaScript functions that are entered in these text boxes are executed when the report section is rendered. For a detailed explanation of the interactions of scripts when rendering refer to Chapter 6, *Scripting with JavaScript*.

After setting the values in the tabbed panes and clicking the Finish button, the section is added under the Sections node of the report template.

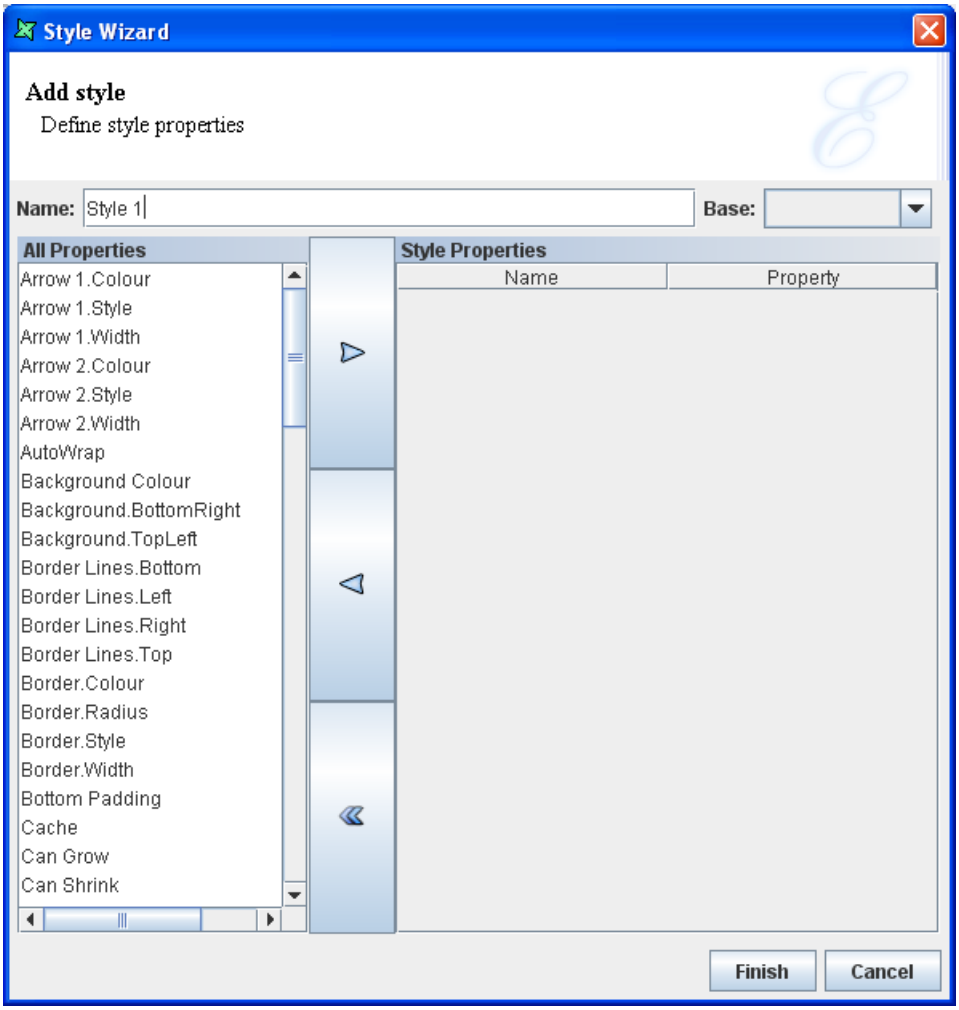
A Section can be copied using the popup menu and pasted into another report template. It can also be pasted into the same report, to make a copy before modifying it.

Styles

Every Report template will have a set of styles. When a new Report template is added, the default style settings will be included for the report elements in the Styles node.

The default style can be edited to suit your requirements. A new Style can be added by selecting Add Style from the Styles popup menu. On selecting Add Style from the popup, a Wizard appears as shown in Figure 3.8, “Style Wizard”.

Figure 3.8. Style Wizard



The name of the Style is entered in the text box. Select the style to be included in the report from the All Properties list box and click the '>' button. The style properties will be added to the Name column and the property can be specified in the value column.

The Base combo box contains the list of styles that have been added to the report template. This is an optional field. If you want to add a new style based on an existing style then select a Base style from the combo box. The new style will inherit the style properties from the base style that has been selected. That is the style properties of the Base style will be included along with the style properties that have been set in the new style. So when the new style that extends the base style is selected for any particular field the style properties of both the base style and the new style will be set for the field. On clicking the Finish button after setting the properties the style will be added to the Styles section.

If you want to edit an existing style then select Edit Style from the popup menu of the Style. A style can be copied from one report template's Styles section and pasted in the another report template Styles section. It can also be pasted into the same report, to make a copy before modifying it.

The order of precedence in checking styles and their description are given below:

Explicit Named Style: An element placed in the report layout first checks if has a style associated with it. If for instance a style has been defined in the master and sub report then the element placed in the sub report first looks for the style name starting with that of the master report and then the sub report. If there is a style with the similar name in the master report or the sub report then that style applies to the fields. This is an explicit named style.

Implicit Style: If there is no associated style then the element looks up whether there is any default style for this kind of element i.e if it is a literal field then it takes the "field-literal" style. See the RML Raw Model document for the style names associated with each kind of component.

Inherited Style: The style associated with the parent of this element. If for example the detail has some style associated with it then all the elements placed in the detail section will get that style.

Finally, if there is no match found, then it makes use of the global style which is fixed as black.

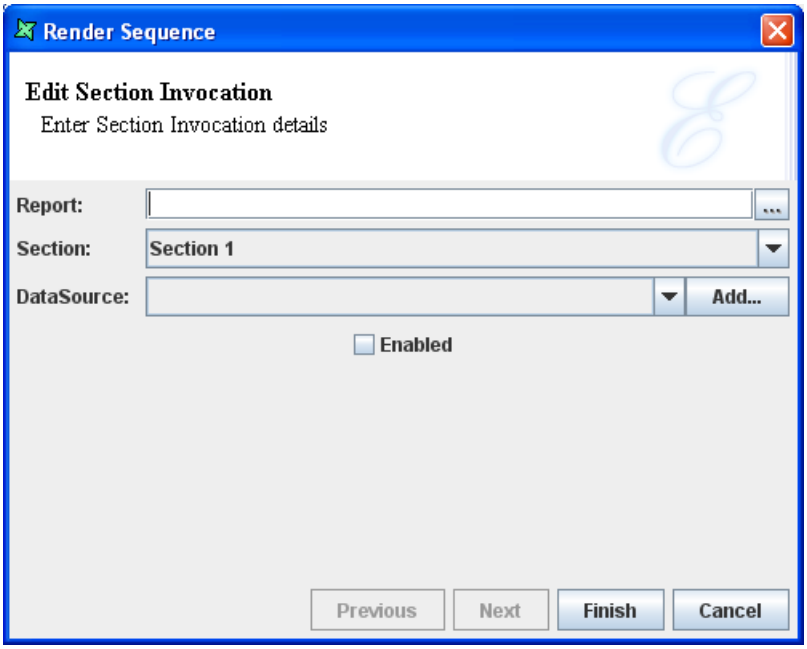
Some styles can be inherited from the parent, others cannot. The styles that can be (defined by CSS) include FontName, FontColor, FontSize, FontBold, FontItalic FontUnderline and FontStrikethrough.

Render Sequence

Initially, when a report template is added to the repository it contains a default section. When a template contains multiple sections, it is the Section Invocation table that controls the order of section rendering. The initial configuration of the Section Invocation details shows the default section is enabled for rendering.

If a new section is added to the report template and this section has to be rendered then click on the New button on the Section Invocation panel. The Render Sequence Wizard appears as shown in Figure 3.9, “Render Sequence”. The render sequence can include sections from both this report template and other reports in the repository. Choose the report (if not the current report) and then select a section name from the combo box. You can also choose to specify a dataSource, which will override any datasource defined with the section itself. This allows the same section to be used repeatedly in the render process, with a different datasource each time.

Figure 3.9. Render Sequence



The Section Invocation details can be edited by double clicking on the column. The Render Sequence Wizard appears in which the values can be altered. On clicking the Finish button the values in the column will be edited.

If there is more than one row specified in the Render Sequence table, then the sequence in which the sections are being rendered can be changed by selecting the column and clicking on the Up/Down arrow button.

Parameters

The Parameters panel is a table containing Name, Value and Enabled columns. The parameter name and value are entered in the corresponding columns of the table. The Enabled column contains a check box which can be selected to enable the parameter.

The parameters are used to control the way a report is rendered. The parameter can also be used to control data like selecting a range of employees from a list of employees.

If the "Propogate data source parameters to report" check box is selected in the Choose DataSource screen of the Report Wizard then any datasource parameters are added to the Parameters panel.

When the Propogate data source parameters to report check box is not selected in the Choose DataSource screen of the Report Wizard while adding a report template then the parameter name and value can be added later. Similarly if you switch over to another data source from the existing data source or add a new data source that may contain parameters then you can add the parameter name and value to the Parameters panel by clicking the New button.

The Parameter details that has been added to the Parameters panel can be edited or deleted by selecting the row and clicking the corresponding button. You are allowed to have multiple parameters with the same name, but only one can be enabled at a time. This allows you to store alternate sets of parameters for different situations without having to re-enter the values.

Layout

The Layout tab was shown earlier in Figure 3.1, “Report Layout”. It contains the Navigation bar at the top, the parts of report such as headers, footers, etc and the markers in the central panel, the toolbar containing the report components on the left and the Data|Shapes|Styles tree panel and the Properties panel on the right.

Navigation Bar

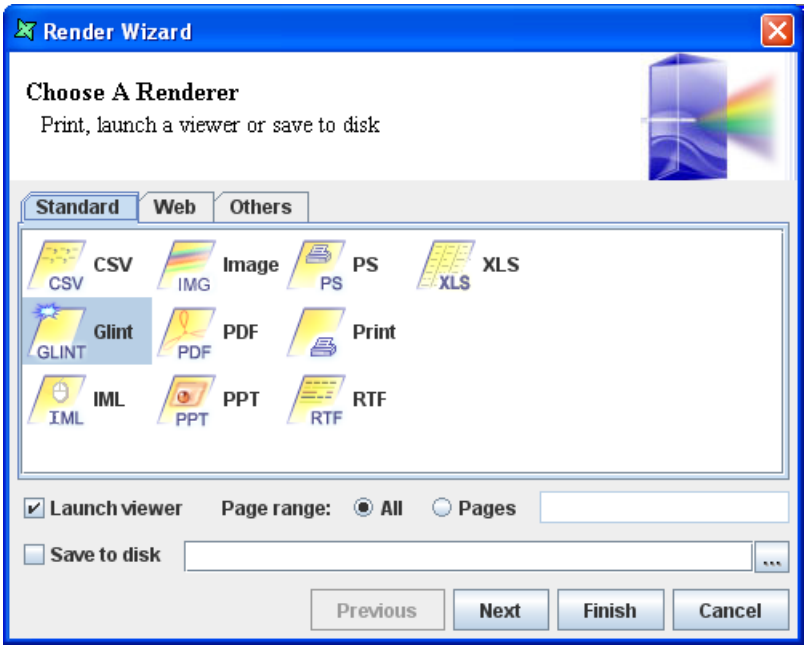
The Navigation Bar contains the Zoom and Section combo box. The Zoom combo box contains the Zoom percentage. You can select the desired Zoom percentage from the combo box based on which the Report parts will be Zoomed.

The Section Combo box contains the list of sections of the report template. You can select the section for which the layout has to be modified.

The `Redo` button changes back the element to the previously changed stage and `Undo` button reverts the element back to the state before the change.

The Render Report button is used to open the Render Wizard, as shown in Figure 3.10, “Render Wizard”, to guide you through the report rendering process. Select an output type from the icons shown and click Next to edit any custom properties of that output type. Alternatively, you can immediately press Finish to use the default properties (or the values selected previously, if you are rendering the same report again). The different output types are explained in detail in Chapter 5, *Report Rendering and Output Formats*.

Figure 3.10. Render Wizard



If you want to repeat a render with the same output type and parameters, you can choose the Repeat Render button, which will by-pass the wizard.

Toolbar

The toolbar contains a set of elements that can be used for adding more features and elegance to the reports. The description and properties of the report elements will be covered in the next chapter on Report Elements.

Horizontal and Vertical Rulers

The horizontal and vertical rulers are used like a scale for page measurements. By default the width of all the headers and footers in the reports layout remains the same.

The height of the headers and footers can be altered in a report. The vertical rulers are used to measure the height of the headers, footer and the detail sections.

The Expand/Collapse button is used to expand or collapse the report parts(headers, footers, etc). When a part is initially in the expanded state click the [button icon] button to collapse it. Then the part get collapsed and you will not be able to work in it. Then if you click the [button icon] button to expand it the parts will get expanded and you can continue working on it.

The markers on the horizontal and vertical rulers are used to insert guidelines. To insert your own guideline, simply click in the horizontal or vertical ruler - a red triangle will appear. To align the fields, drag them to the guidelines. The benefit of using guidelines is that you can move the guideline markers on the ruler and all the elements currently snapped to the guideline will automatically move. For Instance, if you place a horizontal guideline in the report header then drag all the column heading labels to the guideline, you can then reposition the guideline marker to automatically move all the column headings up or down. This saves you from selecting and moving each of the column heading labels manually. .

When the markers are no longer required then they can be removed by opening the marker popup menu and choosing "Remove" or by dragging them to the edge of the report layout.

Report parts

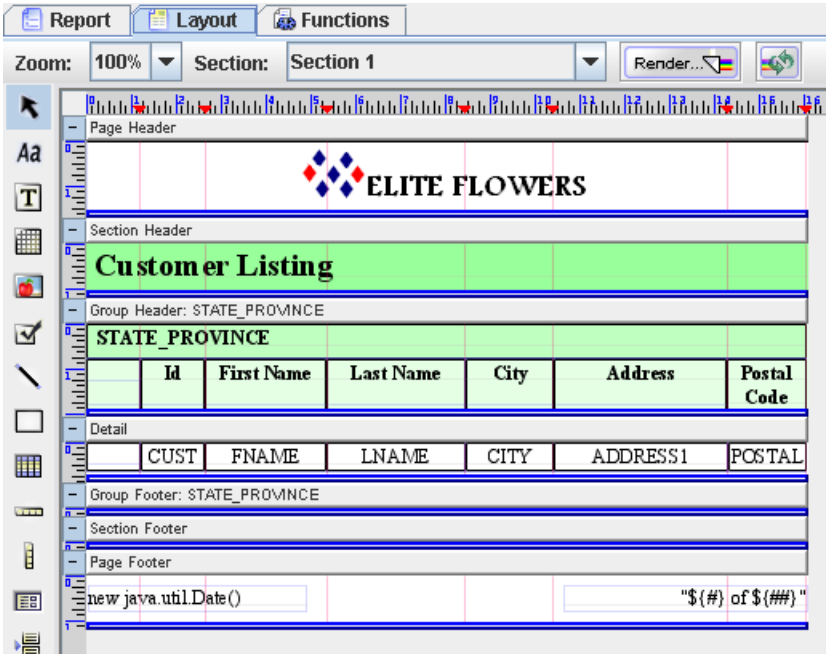
The illustration shown in Figure 3.11, “Sample Layout” depicts a typical layout. Although the particular arrangement of these parts will vary depending on the report type and style you have selected, the part themselves remain constant and can be manipulated in a variety of ways.

The page header typically contains information relevant to the particular page viewed. The page header might contain the page number, column headings, border lines or a bitmap. The text and data of this page is automatically printed after the top margin of every page.

The page footer probably contain information relevant to that particular page viewed and appears at the bottom of every page. It may be used to print the page total or other pertinent text. Page headers and footers can contain static content, but they are more commonly used to display varying content like page numbers or information about the contents of a page.

The section header contains information of the section which will be displayed at the beginning of each section and footer contains information of section and is displayed at the end of each section in the report.

Figure 3.11. Sample Layout



Group header present information about the particular data group being viewed - for instance the group name and a brief description about the data contained in the group and appears at the beginning of a new group of record. Group footers might contain totals or other similar information that appears at the end of each group. Group headers and footers can be added by turning on the corresponding check box when adding the group column in the Sorting and Grouping tab of the Section.

The Detail section contains the main body of the report's data. The detail section prints individual records, and repeats until all records have been printed. You may expand the Detail section to place multiple fields above or below each other.

Field labels name specifies fields displayed in the report. By default Elixir Report Designer uses the field names from the original data source as the field labels in the report, but you can edit the field labels as needed.

Fields are items of data, from the data source specified in your report.

Data|Shapes|Styles tree panel

This panel consists of the the Data tab, Shapes tab and Styles tab.

The Data tab contains the tree view representation of the DataSources that this report references. A new datasource can simply be added under Data tab by drag and drop from Elixir Repository to Data tab.The DataSource fields can be directly selected from the Data tab and placed inside the report layout. The popup menu provides a Refresh option to update the tree if the datasource structure has changed.

The Shapes tab contains the tree view representation of the report parts and the type and names of the fields placed in the layout headers, footers, etc.

The Styles tab contains the tree view representation of the styles of the report parts, fonts, fields placed in the group header, group footer and section label etc. Styles can be added to adjust the report outlook and alignment etc.

Properties Table

The Properties table contains the Name and the Property column. The property name and the corresponding values are listed in the Properties table. When a part of the report or the field placed in the layout is selected then the corresponding properties are listed in the table.

Chapter 4
Report Elements

Overview

This chapter introduces the report elements and the various properties they provide. Properties hold strings representing different kinds of information, such as colours and fonts. The properties of the elements are listed in the Properties table of the Layout tab - some of the most commonly accessed properties are also available on a popup Properties dialog accessed by double-clicking on a report element or by choosing Properties... from its' popup menu.

Colours

Colours in Elixir Report Designer are specified as an rgb string, in the form rgb(R,G,B,A) or by name. The R, G and B values are numbers in the range 0-255 which is the amount of red, green and blue in the colour and A is an alpha value, or transparency control, which is also a number in the range 0 (transparent) to 255 (solid). The alpha value is optional and defaults to solid, so that rgb(100,100,100) is the same as rgb(100,100,100,255). The text rgb(0,0,0) corresponds to the colour black and rgb(255,255,255) corresponds to white. While this format gives you complete flexibility in terms of colour choice, it can be hard to remember the appropriate rgb values. Therefore, as an alternative, you can enter the names of the colours, such as Black, White, even LightSalmon and SteelBlue, if you prefer. Elixir Report Designer supports the standard colour names originally defined by the X Window System and now used in SVG, HTML and CSS. The table of pre-defined colour names is shown in Figure 4.2, “Named Colours”

Figure 4.1. Choose Colour

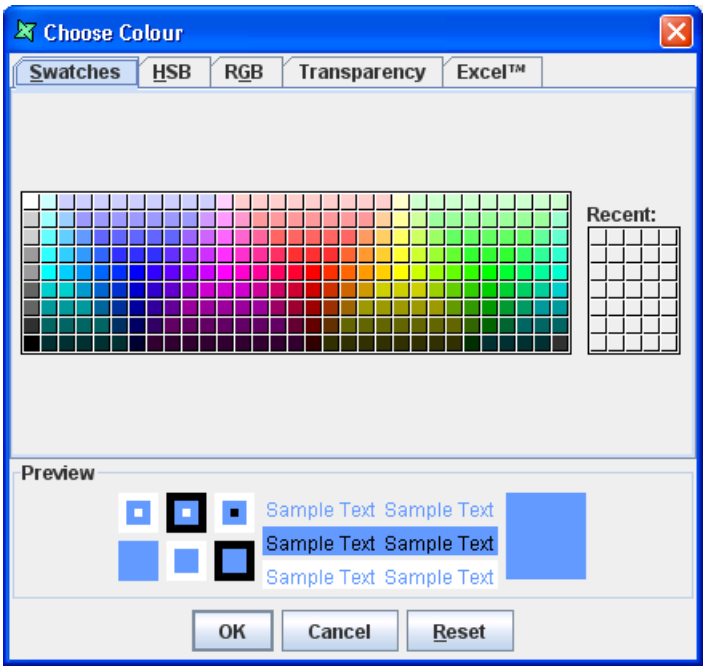
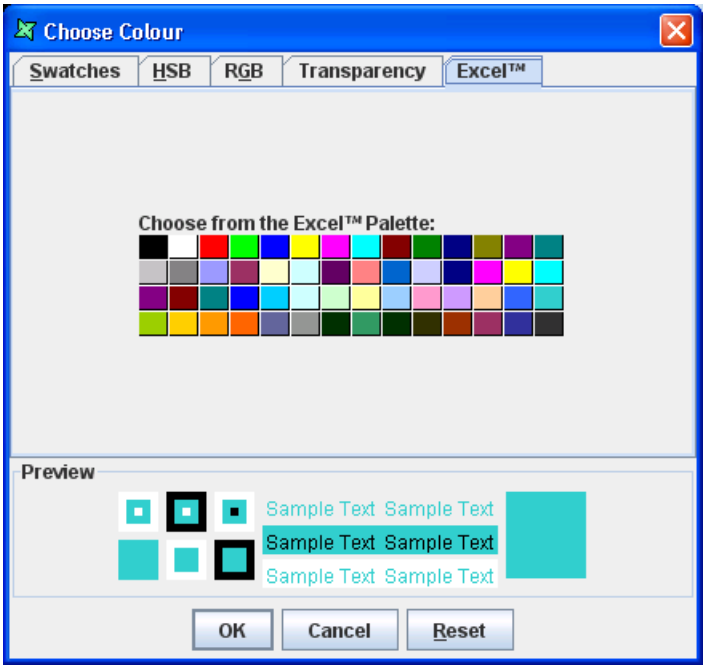


Figure 4.2. Named Colours

Figure 4.3. Choose Colour (Excel Compatible)



A further option is to click the button on the right of each colour field to select from a Choose Colour dialog as shown in Figure 4.1, “Choose Colour”. This dialog also allows the colour to be made partially transparent by adjusting the alpha value on the Transparency tab.

Figure 4.3, “Choose Colour (Excel Compatible)” shows the colours that are compatible with Excel since some colours in the `Swatches` tab does not appear correctly when rendered in Excel.

Common properties

Many elements share some common properties like background colour, border lines, etc. To avoid repetition, the common properties are described here:

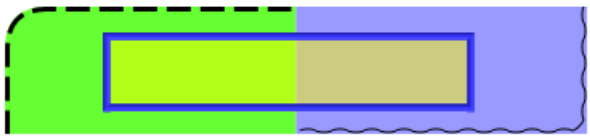
Background Colour

The background colour is used to fill the background of a shape. It may be partially or completely transparent in which case colours will show through from objects behind.

Border Lines

The Bottom, Left, Right and Top border lines can be individually controlled to create interesting effects as shown in Figure 4.4, “Borders and Backgrounds”.

Figure 4.4. Borders and Backgrounds



Borders

Borders are controlled through a number of inter-related properties, colour, width, radius and lines. The border style governs the use of these properties. Borders are rectangular, though they may have curved corners set using the radius property. Styles Groove, Ridge, Inset and Outset don't support curved corners. Double borders are created by dividing the width into three and painting the inside and outside thirds. This means the border width must be at least three pixels (60 twips) and ideally a multiple of three for the Double effect to be visible. Groove, Ridge, Inset and Outset vary the darkness of the border colour to simulate depth. The darkness of Black can't be varied, so these styles won't be visible with a black border. Finally, the Wave style uses the border width to determine the amplitude of oscillation of the line, so it needs a width greater than one to show any effect.

Can Grow

By default, a field will use the space exactly as drawn on the designer. When rendering longer text, it can result in the some characters being lost. To avoid this, enable Can Grow in the Properties table. When this property is enabled the report element grows in height so that all the data it contains is printed properly. Can Grow also applies to details, headers and footers. The container will grow to the height necessary to encompass all of it's children, retaining any required padding. Note that page headers and page footers can neither grow nor shrink.

Can Shrink

Can Shrink provides the opposite facility to Can Grow. When there is no data to be displayed a the field then it is unnecessary to allocate space for that component. By using Can Shrink you allow a component to reduce height to accomodate less (or no) data. Can Shrink also applies to details, headers and footers. The container will shrink to the smallest height to encompass all of it's children, retaining any required padding. Note that page headers and page footers can neither grow nor shrink.

Data

The Data property allows you to select the source of data to fill a Field. This property is connected to a dialog which allows you to choose from DataSource Fields, Operations, Scripts, URLs or Literal values to supply the information to show. The exact choices available vary with the kind of control.

First Line Indent

When a value is specified for this property the first line of the field will be indented based on the specified value and the unit of measurement. If the value given is negative, then the first line will remain flush left and subsequent lines will be indented by the specified value.

Font

The Font properties allow you to change the font name, size, colour and the style of text. While fonts can be set explicitly on individual elements, it is a good idea to define styles for fonts, eg. TitleFont, FooterFont etc. and then reference the styles inside the report elements. This allows changes to values to be made by simply altering a style rather than manually editing every single element.

Keep Together

The Keep Together property is provided by containers are ensures that wherever possible, the contents of the container will be kept together on the same page of the report. When this property is enabled then the data is printed on one page instead of splitting it across two pages i.e if it can't fit in the remaining space on one page, it will advance to the next page and render there. When this check box is turned off it prints as much as data as possible in the first page before advancing to the next page.

Keep With Next

This property applies only to bands (details, headers and footers). If set to true, then it will ensure that the band remains on the same page as the following band. This is useful in cases where you want the header and first detail to always be together - you don't want the header at the bottom of a page and the detail on the next page. If you enable Keep With Next on the header by turning on the check box then, if there is no room on the page for the subsequent detail, it will push both the header and detail onto the top of the next page.

Padding

The Padding properties define the space between the element border and the element content. Top, right, bottom, and left padding values can be set individually. Negative values are not allowed.

Position

This property determines the position of the element in the report layout.

Fill: If the Fill check box is selected then the element resizes itself to Fill the whole container in which it is placed. The Left, Right, Height and Width attributes can be adjusted manually here to give precise positioning which might not be achievable with a mouse.

In addition to the above properties, components which contain text, that is Data Field, Label and Data Grid also support Text align and Vertical Align.

Text Align: The Text Align field values are Left, Right, Center and Justify. The text that is displayed in the field is aligned horizontally based on the value selected from the combo box.

Vertical Align: The Vertical Align field values are Top, Center and Bottom. The text that is displayed in the field is aligned vertically based on the value selected from the combo box.

In the Image element there is a Horizontal Align property. Similar to the text align the image is aligned based on the options selected from the combo box. Unlike text alignment, horizontal alignment does not support justified mode.

Lock Handles: When this option is checked, it will lock the element in position and any resizing is impossible. When the placeholder of the element is grey in colour instead of green, it means that the particular element is locked.

Script

Render If, On Render Begin and On Render End fields can hold JavaScripts which control the rendering of the element. By clicking the button in the fields a larger, syntax-coloured editor is available. For more information on working with scripts refer to Chapter 6, *Scripting with JavaScript*.

Style

The existing styles in the report template are listed in the combo box. You can choose the required style for the element from the combo box.

URL

There are three URL properties which are available for Chart, Data Field, Grid, Image and Label elements. The URL property holds a URL value (eg. http://www.elixirtech.com) that will be used as a hyperlink jump when the report element is clicked in render types that support linking (HTML and PDF).

URL Description

The `URL Description` property provides a description which may be shown as a tooltip, if the viewer supports it.

URL Target

`URL Target` allows the destination of the URL contents to be set. This is primarily for use in web browsers, so you can force the linked document to show in a separate window or frame if you choose. The value of URL Target (if specified) will become the target attribute of the HTML anchor (<a> element). Here is the list possible available URL targets :

Table 4.1. URL Targets

Attribute	Value
_blank	the target URL will open in a new window
_self	the target URL will open in the same frame as it was clicked
_parent	the target URL will open in the parent frameset
_top	the target URL will open in the full body of the window

Visible

When this property is deselected then the field will not be visible in the output of the rendered report.

Types of Elements

All the report elements that are placed in the report layout has a popup menu. The popup menu contains the menu items Properties, Group, UnGroup, Cut, Copy, Paste, Delete and Ordering.

When the Properties menu item is selected for an element the corresponding Properties window appears where the values can be set.

The Group and UnGroup menu items are initially deactivated. When more than one element is chosen using the Selection element these menu items are activated. The Cut, Copy, Paste and Delete are used to perform the edit operations. The Ordering menu item has the Bring Forward, Send Backward, Bring To Front and Send to Back sub menus.

If for example say there are three elements overlapped one above the other. Select the top most element. On selecting Ordering -> Send To Back option from the popup menu of the element the topmost element appears behind the other two elements. Instead, if the Send Backward option is selected then the element is sent one step behind i.e it appears in between the other two elements.

Select the Bottommost element. On selecting Ordering -> Bring to Front option from the popup menu of the element the bottommost element appears in front of the other two elements. Instead if the Bring Forward option is selected then the element is brought one step in front i.e it appears in between the other two elements.

When the values are being edited in the Properties column then the Names of the Properties in the Name column turns to bold indicating that the values of that particular property is being edited.

If a value say for instance value1 has been assigned for a particular property of an element and then we edit the value to value2. When you want to revert back to the original value1 select "Revert to Style" from the popup menu of the respective element. Now the value1 is assigned again and the property's font changes from Bold back to Normal.

If for instance you set the background colour of an element say as green colour. Now, add a style in the report tab and specify a different background colour(blue). Then when you assign this style in the property table of the element the background colour still remains as green. So if you want include the colour specified in the style click in the row corresponding to the background colour in the property table and select revert to style from the popup menu. Now the background colour of the element will be changed to blue.

All the Graphical elements have the Properties Wizard in which the values can be entered. The non-graphical elements such as Horizontal Box, Vertical Box, Selection and the Page Break does not have the Properties Wizard.

Manipulating Report Elements

Elixir Report Designer allows you to select, resize and move report elements such as labels, Data Fields, etc.

Selecting Report Elements

In Elixir Report Designer you can select a single element or multiple report elements at a time.

Selecting an individual report element

1. Click the report element. Green handles will appear on the selected report element.

2. To unselect the report element, click anywhere outside the report element.

If the desired element is a child of another element, for example it is within an hbox, vbox, table or group, then a click will select the parent element. Use control-click to select an element within the currently selected parent. You can see from the shape tree on the right hand side of the designer which element in the tree is currently selected. HBox and VBox elements contain cells, which are not themselves editable. The box contents are within the cells. You will notice that a cell shows grey handles to indicate it cannot be resized.

Selecting multiple report elements

This selection method is used to select multiple report elements in the same section of the report.

Marquee selection

1. Position the cursor near the first report element you are going to select.

2. While holding down the left mouse button, drag it to include all the report elements that you wish to select. When the mouse is dragged a box with blue coloured dotted lines will appear using which you can surround the elements.

3. Release the mouse button and all the report elements will be selected.

Shift Selection

1. Select one report element.

2. While holding down the shift key select the other report element.

To unselect the elements click outside the selection(s).

Moving Report Elements

Moving report elements within its own section

Report elements can be moved by dragging them with the mouse which is fine for rough positioning, or when snapping to the ruler or edge. Alternatively, you can use the cursor keys for finer-grain control after selecting the elements or you can use a combination of command keys+cursor keys for smaller increments. Finally, you could also enter the Left and Top values in the position node of the property table which is the most accurate but time consuming.

Moving report elements between different bands of the report

1. Select the report element(s) and Cut them using the popup menu or the accelerator key **Control**-**X**.

2. Click in the target area (the left hand ruler will highlight to show it is current) and then Paste, using either the popup menu or the accelerator key **Control**-**V**.

Copying Report Elements

1. Select the report element(s) and Copy them using the popup menu or the accelerator key **Control**-**C**.

2. If you are copying the report elements to a different band then click in the target area. The left hand ruler will highlight to show it is current.

3. Paste, using either the popup menu or the accelerator key **Control**-**V**.

Resizing Report elements

Select the report element(s). Move the cursor towards the green handles. The cursor becomes the resize cursor. Now, drag on the resize cursor to increase or decrease the size. Alternatively select the report element(s) and in the properties table change the Width and Height properties.

Selection

The various types of report elements and their individual properties are given below:

The Selection element is used to select the elements placed in the report layout. This is useful when we want to perform a common operation on all the controls. If a number of elements have to edited, aligned, resized, grouped or ungrouped then all the elements have to be selected at a time using the Selection element in the toolbar.

The Align and the Resize menus are included along with the other menu items only in the Selection element's popup menu. Similarly, the Group and UnGroup menus are active only for the Selection control's popup menu.

The submenus of the Align menu are Align Left, Align Right, Align Top and Align Bottom. Select the elements that are to be aligned using the Selection element. On selecting Align -> Align Left, all the elements are aligned inline with the element in the extreme left. When Align -> Align Right, all the elements are aligned inline with the element in the extreme right of the layout.

If Align -> Align Top is selected then all the elements are aligned inline with respect to the top most element in the group. When Align -> Align bottom is selected then all the elements are aligned inline with respect to the bottommost element of the group.

The submenus of the Size menu are the Widest, Narrowest, Tallest and the Shortest. Select the elements that are to be resized using the Selection element. On selecting Size -> Widest the width of all the elements increases in proportion to the width of the widest element in the group. When Size -> Narrowest is selected then all the elements decreases in size in proportion to the narrowest element in the group. When Size -> Tallest is selected then all the elements decrease in size in proportion to the tallest element in the group. If the Size -> Shortest is select then all the elements decrease in size in proportion to the shortest element.

Label

The Label element is used for adding static text such as titles, caption, etc. in Report.

The Field tab consists of the Field Type combo box. The default Field Type for Labels is Literal. The text to be displayed in the Label can be entered in the text field and it will be shown exactly as entered. No formatting will be applied, because you can enter whatever text you want to see. On selecting field types other than Literal the Label element changes to a Data Field element. In these other modes, formatting is applied

Select the Font tab. In this tab the font settings are specified. The Font Name can be selected from the combo box. The Size of the Field is selected from the field and the Font styles such as Italic, Bold, Strikethrough or Underline is chosen by selecting the corresponding check box. The colour of the font is entered in the text box. Alternatively, by clicking the button on the right of the text box the colour and the transparency can be selected from the Choose colour dialog window as mentioned earlier.

As mentioned above, the format tab is not applicable for Literal Labels.

The Scripts tab contains the Render If, On Render Begin and On Render End text fields. For more details on the use of JavaScript refer to Chapter 6, *Scripting with JavaScript*.

After setting all the properties click on the Finish button.

The additional properties that are specific only to the Label element are given below:

Orientation: The degree of orientation is entered in the Orientation field of the Properties table. The values that can be specified for the Orientation are 90, 180, 360 and so on. The text will be tilted based on the angle specified in the Orientation field.

Note

Hide Duplicates is not applicable for the Literal field type, it is only applied for other field types. For a description of Hide Duplicates, please see the Data Field component below.

Data Field

The Data Field is used for dynamic binding of data into a report at runtime. The Data Field can be selected directly from the tool bar and placed in the report layout. Select the Properties option from the Data Field's popup menu.

The different Field types are Field, Operation, Script, URL and Literal.

Field

By default the Field type is a datasource Field. The DataSource is selected from the list of data sources in the combo box. Alternatively, by clicking the Add button on the right of the combo box you can select a data source from the Configure DataSource Wizard. On clicking the Finish button in the Configure DataSource Wizard the selected data source will be added to the list of data sources in the DataSource combo box.

The data source field name and the corresponding data types are listed in a table in the Field tab. Select the required field from the list of DataSource fields. The name of the field that has been selected is displayed in the Expression text box.

Operation

The Operation field type is selected from the combo box. The DataSource is selected from the list of data sources in the combo box. The required field is selected from the list of fields in the table.

The Average, Comma Separated List, Comma Separated Set, Count, First, Last, Maximum, Median, Minimum, Percent, Percent100, PercentCount, PercentCount100, StandardDeviation, Sum, Variance, Year to Date and Month to Date functions are listed in the combo box from which you can selected the required function type. The definitions of all the available operations is given in the Repertoire manual, "Function Reference" chapter.

The Over Group or Over All option is selected depending on whether you want to perform the function calculation over a particular group or on the whole set of data. The combination of Over Group/Over All and Running Value determines the range of records that the operation will apply to. The start of the range is controlled by Over Group and Over All. If Over All is chosen, then the range begins with the very first record of the datasource. If Over Group is chosen, then the range begins with the first record of the current group. Where there are multiple levels of grouping, the current innermost group is used.

If Running value is not selected, then the end of the range is based on the Over All/Over Group value. Over All means the last record of the datasource. Over Group means the last record of the current group. When Running Sum is enabled then the current record is used as the end of the range.

You can calculate running values that accumulate within each group and are reset each time a new group of records begins. You can also calculate running values that accumulate throughout the report. For doing this select Sum function from the combo box, select the Over All/Over Group option and finally select the Running value check box. Similarly the running value for average, maximum, minimum, etc can be calculated.

To illustrate the use of summation and running values, here's a simple example. We start with the data in Figure 4.5, “Sample Data”. We will group it by Name and perform operations on the Value. A report is designed, as shown in Figure 4.6, “Sample Running Sum Report”. The report demonstrates the use of running values and over all/over group.

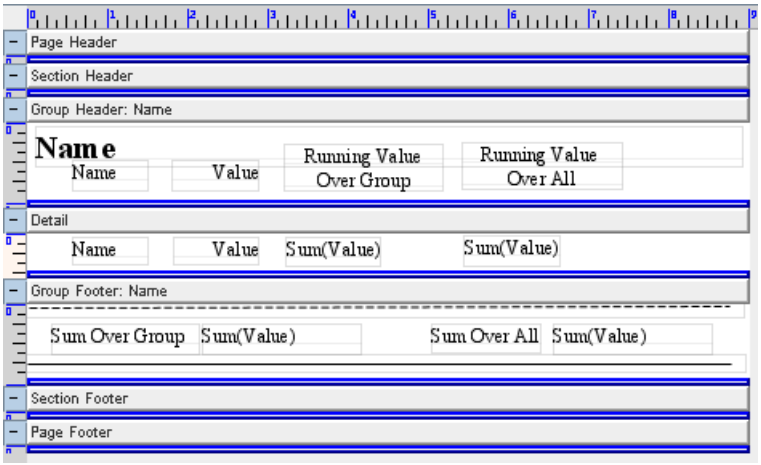
Figure 4.5. Sample Data

Data

[Result]

Name (String)	Value (Integer)
A	1
A	2
A	3
A	4
B	5
B	6
B	7

Figure 4.6. Sample Running Sum Report



When the report is rendered, the output appears as shown in Figure 4.7, “Running Sum Output”. Note that in the first group footer, you can still get Sum Over All, even though all records have not yet been processed. Indeed, you can put this value in the header, before any records have been processed. In the 'B' group, Running Value Over Group is reset to 0, whereas Running Group Over All continues from the value 10.0 that was built by group 'A'.

Figure 4.7. Running Sum Output

A

Name	Value	Running Value Over Group	Running Value Over All
A	1	1.0	1.0
A	2	3.0	3.0
A	3	6.0	6.0
A	4	10.0	10.0

Sum Over Group 10.0 Sum Over All 28.0

B

Name	Value	Running Value Over Group	Running Value Over All
B	5	5.0	15.0
B	6	11.0	21.0
B	7	18.0	28.0

Sum Over Group 18.0 Sum Over All 28.0

Script

The Script field type is selected from the combo box. Enter the JavaScript function in the text field.

URL

Select URL as field type from the combo box. Enter the URL in the text box. Select the URL type from the combo box.

The Font and Scripts tabs are as explained previously.

On selecting the Currency, Percent, Number or Date/Time options from the combo box various fields such as Integer Places, Fractional Places, etc appears.

The Min and Max values for the Integer Places are specified in the fields. Similarly, the Min and Max values for the Fractional places are specified in the corresponding fields.

The Show Grouping Separator has to be selected if required and the Grouping Size has to be entered in the field. For example, if Grouping Separator is selected for the number 1234567 and the Grouping size is entered as 3 it might be formatted as "1,234,567".

The Always Show Decimal Separator check box is selected if required. It affects formatting, only if there might be no digits after the decimal point, such as with a pattern like "#,##0.##", e.g., if the check box is selected then the 3456.00 -> "3456." if deselected, 3456.00 -> "3456".

The Custom Pattern check box is selected if you want to specify your own custom Pattern. After selecting the Custom Pattern check box enter the required pattern in the text box.

When the Date/Time option is selected in the Format Type combo box the Date Format and Time Format combo boxes containing various values appears. The values in the Date and Time Format combo boxes are Short, Medium, Long and Full.

If for instance there is a timestamp field having value Fri Oct 05 12 GMT+8:00 1990 then it can be formatted as Short, Long, Medium and Full by selecting the appropriate type in both the Date and Time Format combo boxes.

Short: Short is completely numeric means it displays Date/Time in Numeric format. When this option is selected the output is 10/05/90 12:00 AM

Long: Long gives a longer value. The output value is October 5, 1990 12:00 AM GMT+8:00.

Medium: This gives limited information about the Date/Time Field. When Medium is selected in the Date Format and Time Format combo boxes the output is Oct 5, 1990 12:00:00 AM.

Full: Full is pretty completely specified i.e. it gives the complete detail. The output value is Friday, October 5, 1990 12:00:00 AM GMT+8:00.

If you want to display the date and time field according to your requirements then select the Custom Pattern check box and enter the desired pattern in the text box.

After entering the Format values a sample showing how the number will be formatted based on the settings will be displayed in the Sample text box. Note the format may be dependent on the Locale

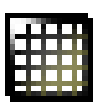
Locale: A Locale property represents a specific geographical, political, or cultural region. An operation that requires a Locale to perform its task is called locale-sensitive and uses the Locale to tailor information for the user. For example, displaying a number is a locale-sensitive operation - the number should be formatted according to the customs/conventions of the user's native country, region, or culture. The desired Locale based on which the formatting has to be done is selected from the property sheet combo box.

Hide Duplicates: When this check box is selected all the duplicate Field values are hidden. For example, if customer_id = 3 gets repeated for more than 1 record then only one field containing customer_id = 3 is displayed and then the record containing a different value is displayed.

Note

Internally, labels and fields are represented by the same component. A label is just a field with a literal data value. No formatting or hide duplicates is applied to literal values. Two icons are shown on the component toolbar to make it easy to select the kind of field that you want, but you can easily change between them.

Data Grid



The Data Grid element is used to display data in tabular form. It is mainly used to handle CJK and other top to bottom languages.

Select a Data Grid element from the tool bar and place it in the report layout. Select the Properties from the pop up menu of the Data Grid element.

Writing mode

This property controls the intrinsic writing direction rendering for a block of content.

The Writing mode combo box consists of values lr-tb, tb-rl, rl-tb and tb-lr. A value is selected from the writing mode to set the direction and flow of text in the Grid element. The default value is lr-tb.

lr-tb: On selecting this the text flows horizontally-from left to right, top to bottom. The next horizontal line is positioned underneath the previous line. This mode is common in western languages.

tb-rl: When this value is selected the text flows vertically from top to bottom, right to left. The next vertical line is positioned to the left of the previous line. This mode is used in Asian writing systems.

rl-tb: When this value is selected from the combo box the text flows horizontally-from right to left, top to bottom. The next horizontal line is positioned underneath the previous line. This mode is used in Arabic and Hebrew writing systems.

tb-lr: When this option is selected the text flows vertically from top to bottom, left to right. The next vertical line is positioned to the right of the previous line. This mode is used in Mongolian writing system.

Glyph

The Glyph values that can be selected are Auto, Inline and Upright.

Auto: When this mode is selected the glyph orientation is determined automatically based on the Unicode character code of the rendered character.

Inline: In this type of mode all glyphs are laid out top to bottom regardless of inherent direction.

Upright: When this mode is selected the glyphs are oriented as if an <angle> of "0deg" had been specified. However all vertical alternates of the glyphs should be used.

Grid Modes

There are two types of grid mode available. They are All and Ideograph.

All

This type of grid can be used to achieve mono-spaced layout. As with 'ideograph', content is divided into strips and each strip is horizontally centered within the smallest number of grid spaces that can contain the grid. The rules for determining strips differs.

Each grapheme cluster with a non-joining base character is a strip. Each non-breakable object (e.g. an image) is a strip. Each run of grapheme clusters with joining base characters that join to each other is a strip.

Note

A grapheme cluster is what a language user considers to be a character or a basic unit of the language.

Ideograph

Content is divided into units called as strips. Each strip is horizontally centered within the smallest number of grid spaces that contain the strip.

Each grapheme cluster with a wide base character is a strip. Each grapheme cluster with a narrow kana character as its base is a strip. Each non-breakable object (e.g. an image) is a strip. Other grapheme clusters are treated as a single strip bounded by the strips described prior. That single strip may be decomposed in several strips if line breaking occurs within it.

The strips are arranged in the grid as follows:

- Each strip corresponding to wide base character that can fit within a single grid space is rendered in the horizontal center of the grid space.
- Each strip corresponding to other characters are placed in the center of the smallest number of grid spaces necessary for it to fit. If a line break occurs within such a strip, the strip is treated as two or more separate strips whose individual placement follows the same rules as those for a single strip.
- Strips corresponding to non-breakable objects and strips corresponding to wide base character that are wider than a single grid space, are each centered within the smallest number or grid cells necessary for them to fit.

The 'ideograph' mode disables all special text justification and glyph width adjustment normally applied to the contents of the block element. If a line break opportunity cannot be found in a text run going over the line boundary, then that text run will be pushed down to the next line and the last part of the previous line will be left blank.

Text Height

The text height defines the block-progression value for inline boxes. Depending on the content size, one or a multiple of 'line-height' will be necessary to accomodate a given inline box.

Text height is font ascent+descent where font ascent indicates the average height of the font from the baseline to up and font descent indicates the average height of the font from baseline to down. So the total height of the font is font ascent+descent.

Line Height

The line height is the number of pixels taken up by a line of text, from the baseline of one line to the baseline of the next. The line height is automatically set equal to the font-height.

The Line height is font ascent+descent+leading. The font ascent and descent are as previously explained. The font leading is the distance between two lines of text. When this option is selected it will be seen that there will be an increase of one horizontal box in about fifteen with a default grid.

Box Count

This option indicates the number of boxes that fits in the Data Grid element. Select this option and enter the values in the Across and Down Fields. The value entered in the fields indicates the number of boxes to be placed across and the number of boxes to be placed downwards inside the Grid element.

Fixed Width

When a value is specified in the Fixed width text box each cell will have the specified width. The Unit of Measurement can be selected from the combo box.

The other tabs such as Field, Font, Format and Scripts are similar to that as explained in the previous sections.

The other properties that are specific to the Data Grid element are:

Progression: The Progression property in the property table of the Data Grid Wizard is to set the properties such as Text Height, Line Height, Box Count and Fixed width similar to those in the Property Wizard.

By default the progression value is a number which corresponds to the Fixed width. If for instance the number is entered as 20 then all the cells will have the specified width. The other values are Text-Height, Line-Height, Box-Count # # where # is any number. The first # corresponds to the number of cells that are to be displayed across and the second # corresponds to the number of cells that are to be displayed downwards.

Image

The Image element is used to embed an image in the report. The Image element is a rectangular portion into which picture files can be loaded. Elixir Report Designer support image formats such as jpg, png, bmp and gif. The exact file formats supported varies with Java version and vendor implementation. The jpg and png file formats are supported across all implementations, so they are the best choice. The bmp file format is supported in Sun's Java 5 (1.5) and later.

Select a Image element from the tool bar and place it in the report layout. Select the Properties from the pop up menu of the Image element.

Image Type

You can either embed a static or dynamic image in Elixir Report Designer. Static image is a prefixed image that is embedded at design time. For example, company.jpg. This can be achieved by selecting the URL Image type option.

A dynamic image allow a dynamic display of your images based on the data field name passed to in, it is like the Data field element except that it renders image instead of text. An example of dynamic images is a list of photo of the employee in the company. This can be achieved by using Field or Script Image type options.

Javascript graphics can also be used by defining in `Javascript graphics` tab.

The image types and their description are given below:

URL:

On selecting URL from the Image Type combo box the URL text box and the URL Type combo box appears.

The URL path is entered in the text box and the URL type is selected from the combo box. Either a local URL e.g. file:/C:/apple.gif or repository:/EnsembleWorkspace/berry.gif can be specified or a web URL link such as `http://www.google.com/images/logo.gif` can be entered.

Field:

Select the Field option from the Image Type combo box. The DataSource is selected from the list of data sources in the combo box. Alternatively, by clicking the Add button on the right of the combo box you can select a data source from the Configure DataSource Wizard. On clicking the Finish button in the Configure DataSource Wizard the selected data source will be added to the list of data sources in the DataSource combo box.

The data source field name and the corresponding data types are listed in a table in the Field tab. Select the required field from the list of DataSource fields. The field will retrieve the image from the database.

Script

Scripts can be used to specify a URL or path that points to the image. This is used for the dynamic display of images when the report is rendered.

Additionally, there is a Scripts tab contains the Render If, On Render Begin and the Render End text fields in which the JavaScript function to be rendered is entered.

In addition to some of the common properties the Image element has the Size mode property. The different size modes and their description are given below.

Clip: This is the default image mode. When this mode is selected the image is stretched to fit the specified width and height.

If the area(width and height) of the image is greater than that of the image element then the image is fitted in the middle of the specified boundary. Areas that exceed the boundary will be clipped off.

If the area of the image is lesser than that of the image element then the image will be fitted in the middle of the specified area.

Stretch: When you want to resize the image this mode is selected. The aspect ratio will not be taken into consideration while resizing the picture. Depending on the size specified the image will either be shrunk or expanded to fit into the specified area. It should be noted that this function works best with larger images.

If the area of the image is greater than that of the image element then the image is shrunk to fit the area specified, unlike clip where the image is clipped.

When the image area is lesser than that of the image element then the image is expanded to fit the area. If the image element's area is increased invariably then the image is stretched further to fit into the area and thus loosing it's original shape.

Zoom:

When you want to Zoom the image this mode has to be selected. The aspect ratio is taken into consideration while resizing the image.

If the image area is greater than that of the image element then the image is shrunk to fit into the boundary but it should be noted that not the whole area is filled as the image is shrunk while maintaining the aspect ratio.

When the image area is smaller than that of the image element the image is expanded to fit into the boundary but again the whole area is not filled.

Check Box

The Check box element accepts boolean value fields and renders one of two images, depending on whether the value is true (the "on" image is shown) or false (the "off" image is shown). The default on and off images show the two states of a checkbox, but you can use any images to represent the alternate states - for example a tick/cross, a red/green or even a happy/sad combination!

Line

A line is a shape control similar to that of the rectangle element. It is used to draw a line in the report.

Select and place a line element in the report layout. The Line properties are shown. The Line tab contains three panels the Line, Arrow 1 and Arrow 2 and a Preview window.

The Style combo box consists of the various line style such as Solid, Dotted, Dashed, etc. The colour of the line is specified in the text box. You can specify the colour as red, blue, etc or as rgb(R,G,B,a). Alternatively by clicking the button on the right of the text box the colour and transparency can be selected from the choose colour dialog window.

The Arrow 1 and Arrow 2 panels also consists of the Style combo box, the width and color combo box. There are various styles such as arrow, diamond, open arrow,etc.

Rectangle

The Rectangle element is used to draw rectangles in report. All the properties of the Rectangle have been discussed in the Common Properties section. Note that you can turn the four sides of the rectangle on and off independently, so this is a good component to use to ensure your lines are horizontal or vertical. The Line shape is not supported on certain output types, because they don't support arbitrary shape rendering (eg. HTML) - where you only need horizontal or vertical lines you might prefer to use a Rectangle with some sides turned off, because HTML can display these.

Table

The Table element allows you to view data in a table. The table is a container element similar to the Horizontal box and Vertical box. The table element displays data from data sources other than the primary one.

A table consists of three parts. They are the Header, Body and Footer. The DataSource fields are placed inside the Body while the headers can contain column headings, etc. Similarly, footers can also contain any other related details. When a table is rendered, it fetches the table data source and renders one table row for each record in that data source. The fields of the section data source will remain constant as the table is iterating through the records in it's data source, without incrementing the section record index.

Table elements share some common properties with Sub-report. For instance, a table can be used to extract values of a particular customer for example the table can be used to display records corresponding to customer_id=n where 'n' is any number.

The Table Wizard appears as shown in Figure 4.8, “Table Wizard”.

Figure 4.8. Table Wizard

The wizard contains Table and Scripts tabs. The DataSource name is selected from the combo box. The "Show Header/Footer if empty" check box controls whether the header and footer should be shown if the chosen datasource contains no records.

OverAll: All records in the datasource are used.

OverGroup: Records will be added starting with the current record index, up to the next end of a group.

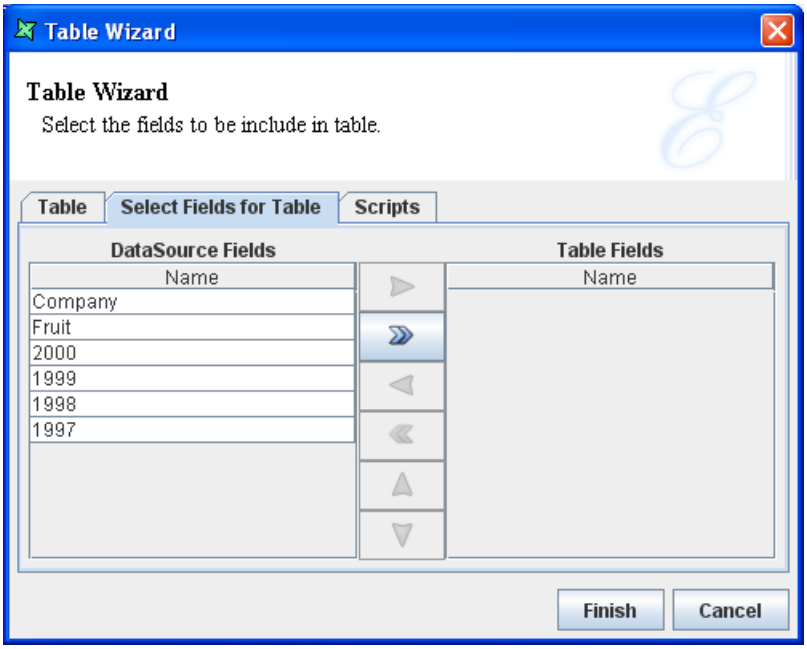
OverCurrentRecord: Only the current record is used. If the table is in a Group header or footer then the first or last record of the group is used. If the table is in a section header or footer then the first or last record of the data source is used.

Usually a different datasource will be used from the section datasource. In this case the record index in the table datasource starts at zero. However, if the table datasource is the same as the section datasource then the table index starts with the current record index in the section datasource.

Once a table has been created, the header body and footer are usually populated with horizontal boxes. You can then append cells to the horizontal boxes to get the desired grid structure and then start adding other components to the grid. Tables in Elixir Report Designer are very flexible - within a table you can place any kind of visual component, including fields, images, barcodes, even other tables.

Fields can be automatically added to the table on creation by choosing the second tab on the wizard as shown in Figure 4.9, “Select Fields for Table”. When this option is chosen, header labels and fields will be added to hboxes within the table sections to speed up creation of the most common table styles.

Figure 4.9. Select Fields for Table



Note

A table should usually use a secondary datasource for accessing records. If the section datasource is chosen then records processed by the table will advance the current record index so that records processed by the table will no longer be available for sequential detail processing.

Horizontal Box

The Horizontal box can be used as a container of elements. The Horizontal box consists of cells. Each cell can contain only one element. The elements that are added will be placed in a row. In the Horizontal box each component is expanded to the height of the container and the components are arranged from left to right in a horizontal manner.

We can also append, insert or remove cells. If you select a cell and click Append Cell from the popup menu the cell will get appended after the selected cell. If you select a cell and click Insert cell from the popup menu the cells are inserted before the selected cell.

In addition to the properties specified in the Common properties section the element exhibits the following individual properties:

Weight: In the property table of the cell the Weight property determines the ratio of size of each cell.

Vertical Box

The Vertical Box element is also used as a container of elements. Similar to that of the Horizontal box the Vertical box also consists of cells and every element is placed inside a cell each. In the Vertical box each component is expanded to the width of the container and the components are laid out from top to bottom.

Similar to the Horizontal Box we can append, insert or remove cells in the Vertical Box. Apart from the common properties discussed above the element also exhibits the Weight property.

Page Break

In reports you can use the Page Break element to mark where you want to start a new page within a report section. The Page Break element is used to insert breaks in between the report pages wherever required. It is useful while creating longer reports.

Sub-Report

A report inside another report is known as sub-report. It is seen that one-to-many relationships are displayed by sub-reports where the main report usually displays one side of the relationship while the sub-report displays the many side of the relationship. There can be a single main report on which multiple subreports can be displayed. A same sub-report can be shared used across any number of main reports.

The Sub-report Wizard allows three values to be entered:

- Report, which may be left blank to indicate the current report
- Section, which should always be defined
- DataSource, which may be left blank to indicate the default datasource defined by the chosen section.

Sub-reports can be set to grow vertically to accomodate all the details that need to be shown, but will not grow horizontally. You need to make sure that the subreport is wide enough to show the full width of the subreport layout.

When a sub-report section is shown within a master report, you may choose to override certain characteristics. For example, if the master report has defined a style for fields, then the sub-report will use it instead of it's own. Similarly, if the sub-report section uses a datasource called MySubDataSource, then the master can define a new MySubDataSource that will override it. While rendering the sub-report, whenever any resources are required and they are identified by name (eg. DataSource, PageSetup, Style etc.) then the renderer will first check the master report for a value, before checking the sub-report. Effectively, the master can be used to override the sub-report values. This allows the same sub-report to be used in multiple contexts and take on the characteristics of the master in each case.

One side-effect of the above behaviour is that, if you have a subreport which uses a page setup called "Default" (the default name) and your master report has a page setup also called "Default", then the master page setup will override the one defined in the SubReport. You should therefore check carefully that datasource names, page setup names and style names in your subreport are unique, otherwise the subreport when embedded will not look the same as the subreport if rendered separately.

The subreport may include multiple columns, but the report engine can only embed multiple columns if the page setup is defined as Across Then Down. Down Then Across columns cannot be handled by the subreport renderer.

Note

If a sub-report and a table is placed in the report layout and they access data from the same data source then they are both considered as discrete component and hence the data source is loaded each time.

Sub Report example

Before creating a main report and a Sub-Report you have to add two data sources one for the main report say customers.ds(customer table) and the other for the sub-report say sales by customerid.ds(sales table) which contains a parameter custid used for filtering sales details based on customer_id. So enter a query in the SQL tab of sales as select * from sales_fact_1997 where customer_id=${cust_id);. Here's how to create a Sub-Report and include it in the main report.

1. Add a Blank report by following the procedure mentioned earlier and selecting sales by customerid.ds as the default data source which points to the sales table of the Mondrian database. While choosing the data source in the Report Wizard must confirm that the Propogate datasource parameters to report check box is turned on. So after the report template has been added it will be seen that the Parameter panel will contain a column with Name - cust_id, Value - ${cust_id} and the Enabled check box turned on.

2. Select section1 and invoke the Section Wizard. Select Sorting and Grouping tab. Click on the Add button. In the Sort dialog window that appears select customer_id from the Name combo box, Ascending from the Sort Order combo box and Each Value from the Group On combo box. Select Show Group Header and Show Group Footer check box and click on the Ok button. Finally, click on the Finish button in the Section Wizard.

3. Select the Layout tab. Place a Horizontal Box element in the Section header. Append another two cells to it.

4. Select and place Label element in each cell and enter Customer_id, Store_id, Unit_Sales and Sales in the text field of the corresponding label element's field tab.

5. Select and place a Horizontal Box element in the Detail section and append another two cells to it.

6. Select and place a Data Field element in each cell. In the Properties Wizard that appears select customer_id from list of field. Similarly add the Data Field element and select Store_id, unit_sales and store_sales from the properties wizard of the corresponding elements.

7. Select and place a Horizontal Box element in the Group Footer:customer_id. In the first cell place a Label element and enter sum(sales) in the text field of the Properties Wizard and click on the Finish button. Place a Data Field in the next cell and in the Properties Wizard select Script from the Field Type combo box. Enter the following in the text field and click on the Finish button.

   ```
   Sum_Sales=Data.getSum("store_sales").getValueOverGroup();
   ```

8. Add a Blank report by selecting customers.ds as the default data source which points to the customer data source of the Mondrian database. Add another data source say Customer_Filter Subreport pointing to the sales by customer_id data source. After selecting the data source click on the next button. Enter "=customer_id" in the value column corresponding to "cust_id" in the Name column and click the Finish button.

9. Select the section section1 and invoke the Section Wizard. Select Sorting and Grouping tab. Click on the Add button and select Customer_id from the Name combo box, Ascending from the Sort Order combo box and Each Value from the Group On combo box. Select Show Group Header and Show Group Footer check box.

10. Select and place a Label element in the GroupHeader: customer_id and enter Sales Report in the text field of the Properties Wizard. Below it place a Vertical box element and append 5 cells to it. Place a horizontal box in each cell. Place label element in each cell of the Horizontal box element and enter Customer_id, Address, Postal_Code, Phone_No, City, State and Country in the text fields of the Properties wizard of corresponding Label element. Similarly, place Data field element in the second cell of each Horizontal box and select customer_id, address1, postal_code, phone1, city, state_province and country fields.

11. Select and place a Sub-Report element in the Detail section. The SubReport Properties Wizard appears. In this select the report Sales by Customer(Sub-report).rml from the Choose a Report dialog by clicking the button on the right of the text box. Select section1 from the section combo box and select Customer_Filter Subreport from the DataSource combo box.

12. Select and place a Horizontal box element below the SubReport element. In the first cell add a label Sum_Sales*.0.5. Place a Data Field element in the second cell of the box. In the Wizard select the Script Field type and enter Sum_Sales*0.5; and click on the Finish button.

13. Finally, invoke the Group Header Properties Wizard and enter Sum_Sales=0; in the OnRenderBegin text field of the Scripts tab. The layout appears as shown in Figure 4.10, “SubReport Example Layout”.

14. On rendering the report appears as shown in Figure 4.11, “SubReport Output”.

Figure 4.10. SubReport Example Layout

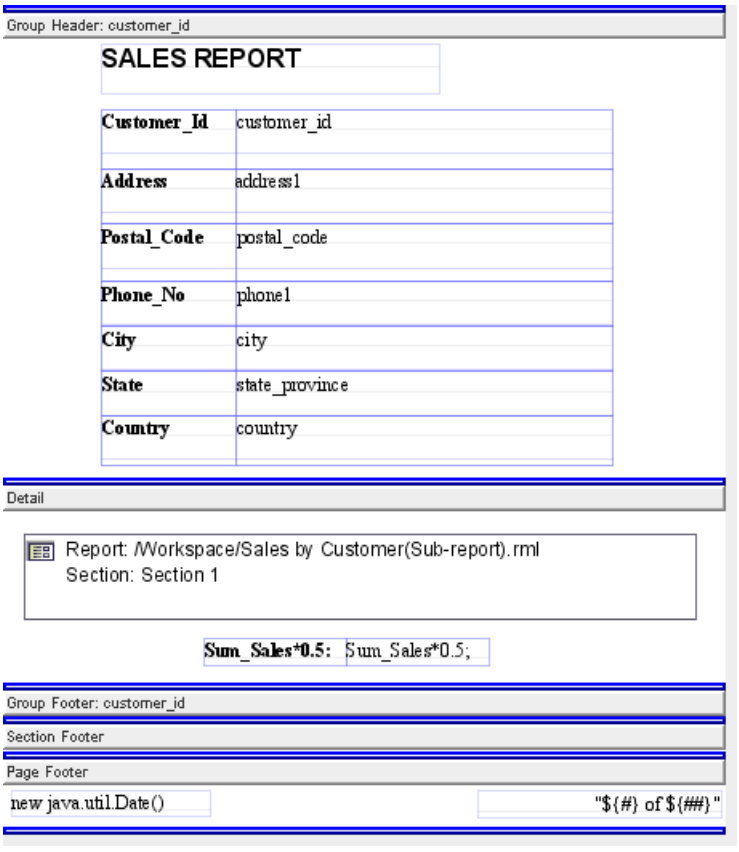
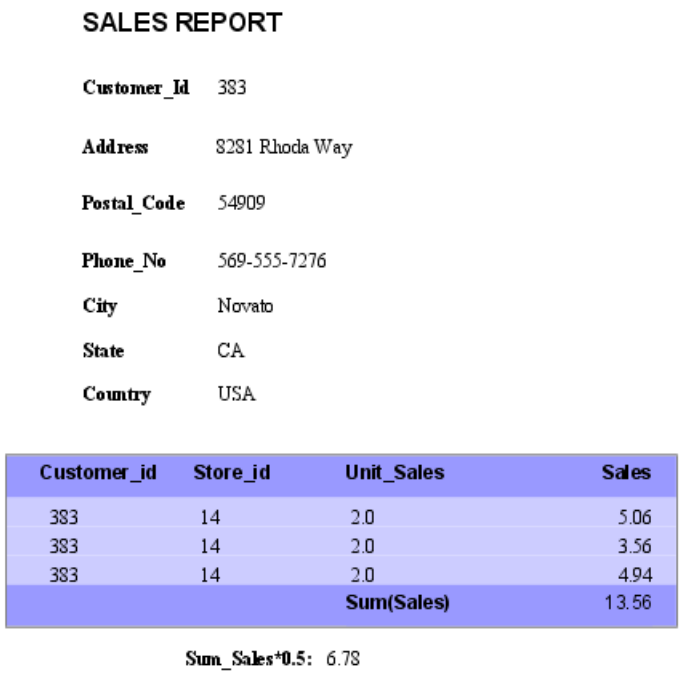


Figure 4.11. SubReport Output



Barcode

A barcode is a series of vertical black and white stripes that are read by a barcode scanner. The vertical black and white lines can contain product information, such as price, weight and size. Once it has been scanned the barcode is translated and often printed out onto a receipt.

There are different types of barcodes known as barcode symbologies. Some symbologies are generic, others are designed for the unique needs of a specific industry.

Barcode Types

The various barcodes and their uses are listed below:

Code 128: Code 128 is a versatile, high-density, variable-length barcode that accommodates letters, number and a variety of other ASCII characters. This type of barcode widely used by the shipping industry.

Code 25: This barcode is a simple, variable length symbology that includes numbers 0-9 and an optional check digit. The barcode types beloging to this category include Interleaved Code 25 and Interleaved Code 25: 2 to 1. Interleaved code 25 is a special type of Code 25 that is self-checking and it is very compact so it does not need as much space as simple Code 25. Code 25 is widely used for inventory and warehousing purposes and is commonly found on photo finishing envelopes, airline tickets, baggage tags, shipping labels etc.

Code 39: This is an alphanumeric code. The types belonging to this category includes Code 39, Code 39: 2 to 1, Extended Code 39 and Extended Code 39: 2 to 1. Extended Code 39 is an extended version of Code 39 that includes the ASCII character set. So with this Code 39 Extended you can also code

the 26 lower characters(a-z) and the special characters you have on your keyboard. Code 39 is used in video rental stores, on identification(ID) cards, and for labels.

EAN: The EAN(European Article Numbering) barcode is used mainly in European countries. The types that belongs to this category are EAN-8 and EAN-13. EAN-8 is a short form of EAN-13. This code is only used if the article is too short for the EAN-13 code. These barcodes can carry numerical information only. These barcodes are primarily used for marking retail items, although it can also be used for in-house applications.

Others: On selecting Others on the left side window of the Barcode Wizard screen the MSI and the Codebar types are listed. MSI also known as Modified Plessey code.

- Codebar: This is a discrete general-purpose barcode with a symbology that accommodates numbers and six other frequently used characters. Codebar is extremely reliable, easy to scan, and highly tolerant of minor printing imperfections. For these reasons Codebar is proven to be ideal for information processing and is used by retail trader for price labelling, libraries, blood banks(labelling blood bags), photo finishing labs, overnight shippers, the military(particularly their supply systems) and many others.

- MSI: This code can display only number 0-9 of fixed length. This MSI code is primarily used to mark retail shelves for inventory control, tracking publications, coding ID cards and so on.

- QR Code: This is a matrix code (or 2-dimensional barcode) created to allow contents to be decoded at high speed. QR Codes are very common in Japan.

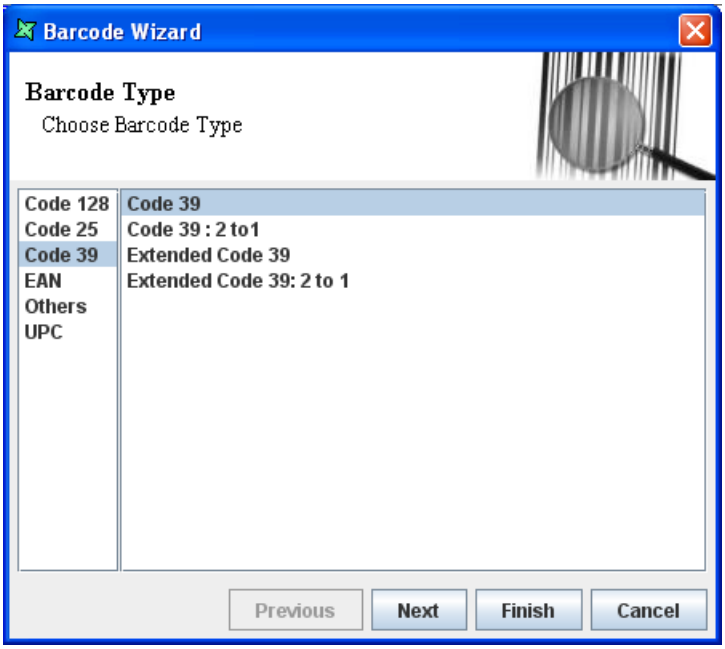
- RM4SCC: This code is created for automated mail sorting processes. It usually encodes a postcode, Delivery Point Suffix (DPS) as well as a check sum character. There are strict guidelines governing usage of these barcodes, which allow maximum readability by machines.

UPC: The UPC/EAN are the most widely used barcodes. The two types of UPC barcodes are UPC-A and UPC-E. The UPC-A code is a standard version of the UPC code and has 12 digits. It is also known as UPC 12 and is very similar to the EAN code. The UPC-E is a very short version with 8 digits, always starting with a zero. The UPC code is a numeric code which is able to display digits 0-9. Each character consists of two lines and two spaces.

The UPC(Universal Product Code) symbols are used on compact disks(CDs), grocery items and magazines, etc.

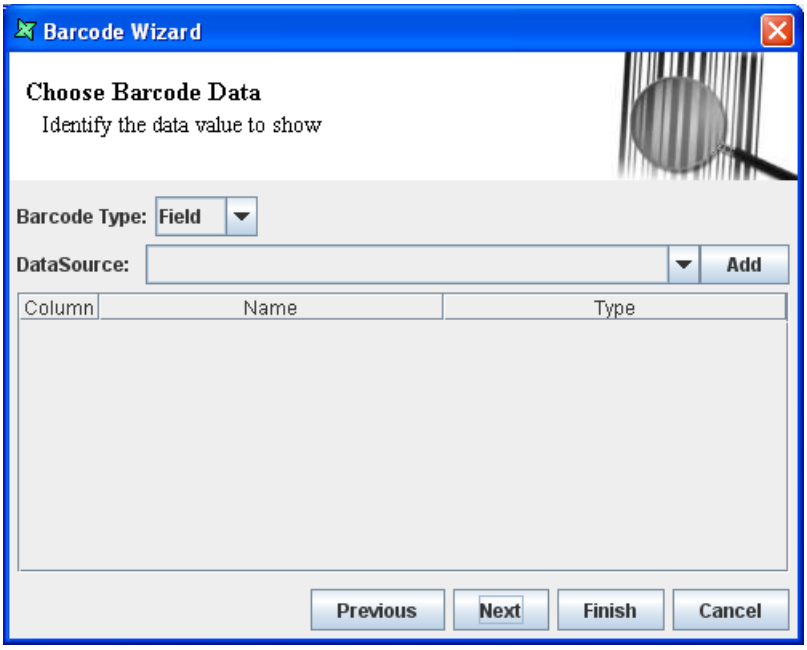
Select the Barcode element and place it in the report layout. The Barcode Wizard appears as shown in Figure 4.12, “Barcode Wizard”. In the Choose Barcode Type screen the different types of barcodes are listed. Choosing from the categories of barcodes on the left panel shows the barcode types of that category on the right panel.

Figure 4.12. Barcode Wizard



After selecting the Barcode type, click on the Next button. The screen appears as shown in Figure 4.12, “Barcode Wizard”. You can select the desired field type.

Figure 4.13. Barcode Control Source



By default, Field is the default Barcode type. Select the DataSource from the combo box. The data source field name and the corresponding data types are listed in a table in the Field tab. Select the required field from the list of DataSource fields.

If you want to display barcode label select Literal from the CheckBox type. The value is entered in the text field. Select the Script Field type and enter the Script functions in the text field.

On clicking Next, the Set Barcode Information screen appears as shown in Figure 4.14, “Barcode Options”.

Figure 4.14. Barcode Options

Height: The value for the height is entered in the text box. Select the desired unit of measurement from the combo box.

Bar Width: The value for Bar Width is entered in the text box and the desired unit of measurement is select from the combo box.

Line Angle: The barcode is rotated clockwise through an angle specified in the barcode. The angle should be entered in degrees

Check Digit: This check box when selected allows you to encode the check digit in the barcode.

Show text: The Show text check box when shows the text version of the barcode in addition to the bars.

Preview Barcode: This panel contains a preview of the Barcode based on the current property settings.

SVG

The SVG is an acronym of Scalable Vector Graphics. A proposed format by the World Wide Web Consortium (W3C) for web page graphics based on vectors, rather than bitmap formats. It defines graphics in XML format. These files will normally be smaller than bitmap files and will scale to different size screens. They do not lose any quality if zoomed or resized. Every element and attribute in the SVG file can be animated.

SVG consists of XML-based file format and a programming API for graphical applications. The key features include shapes, text and embedded raster graphics, with many different painting styles. Elixir Report Designer provides support for Scalable Vector Graphics throughout the SVG element.

The SVG element provides preprocessing capability to enhance the graphics generation, by allowing insertion of field names, running of report script in the report elements in addition of displaying static SVG.

Select the SVG element and place it in the report layout. The SVG can be entered either by specifying the URL or entering the script for embedding the XML file directly in the text field.

The URL path is entered in the text box and the URL type is selected from the combo box. Either a local URL e.g. file:/C:/henryV.svg or repository:/Reports/henryV.svg can be specified or a web URL link such as http://www.w3.org/Graphics/SVG/Test/20010127/toc-sv.svg can be entered.

In the SVG element the data can be embedded by entering

```
<text x="a" y="b">${=Data.getString("Fruit")}</text>
```

Where a and b corresponds to the position coordinates. The data can also be embedded directly by specifying

```
<text x="a" y="b" >${Fruit}</text>
```

After including the text tag in the SVG code you can turn on the Dynamic check box in the property table of the SVG element so that the data source field value will be generated at run time.

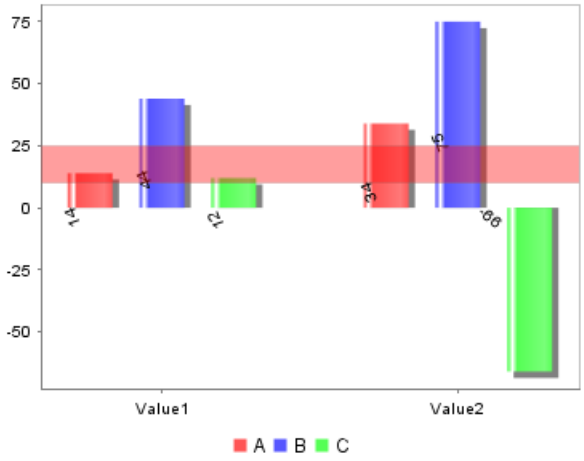
Charts

Charts are visually appealing and make it easy for users to see comparisons, patterns, and trends in data. For instance, rather than having to analyze several columns of data reports, you can see at a glance whether sales are falling or rising over quarterly periods, or how the actual sales compared to the projected sales. Elixir Report Designer provides support for charts through the Chart element.

Note

- Colour tab for customizing the pattern of the chart is available in Pie, Bar, Area, Column, Line, Polar, Waterfall, XY, Bubble and Heat Map Chart.
- The label for Area, Bar, Column, Line and XY charts can be configured to rotate through the wizard tab where plot background is configured. The degree of rotation and direction can be configured in the same page.
- Interval range can be drawn on Area, Bar, Column and Line Chart. This can be done in the Chart Wizard, in the page where the `Key`, `Values`, etc are selected.
- Position of values in Chart element can be customized according to preference. However, this is only applicable to Area, Bar, Column, Line and XY Chart (except Heat Map). The font, color and format of the values can all be customized as well. The prerequisite is to check `Value` in the `Column` tab.

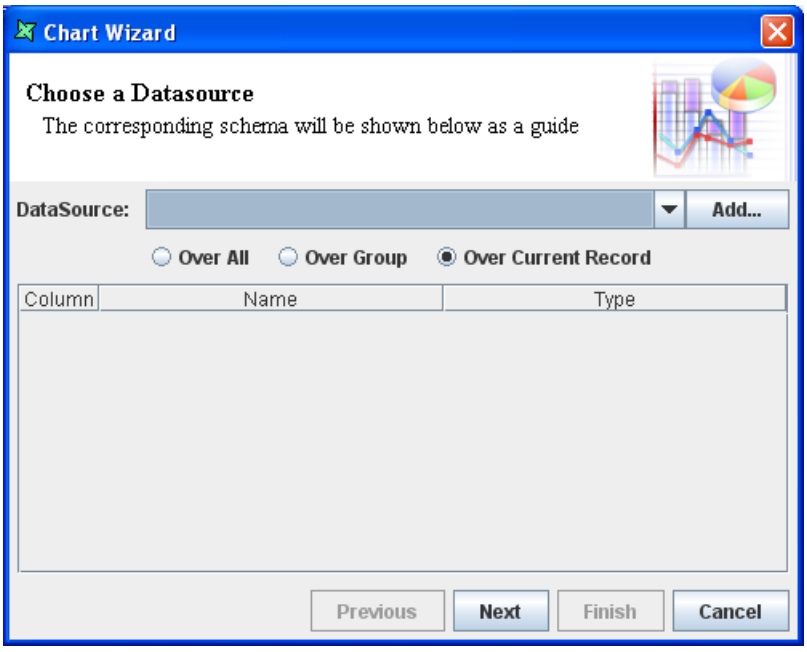
Figure 4.15. Positive and Negative Values Positioning with Interval Marker from 10 to 25



- Area, Bar, Column, Line, XY and Composite Charts can be Domain Gridlines and/or Range Gridlines enabled. Domain Gridlines are lines extended from the points on the X axis. Range Gridlines are lines extended from each respective values on the Y axis.

Select and place a Chart element in the report layout. The Wizard appears as shown in Figure 4.16, “Chart Wizard”. Select the DataSource from the combo box. Alternatively, by clicking the Add button on the right of the combo box select the DataSource from the Configure DataSource Wizard.

Figure 4.16. Chart Wizard



OverAll: On selecting this option, the whole set of records are taken into consideration irrespective of the grouping.

OverGroup: If this option is selected then the plotting is done based on the grouping of data is considered.

OverCurrentRecord: If this option is selected for the chart in the Group header, then the first record of the group is considered. If this option is selected for the section footer, then the last record of the

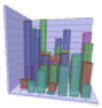
data source is considered as the current record or if this option is selected for the section header then the first record of the data source is considered.

Enter the Display Name if required.

Click on the Next button to navigate to the next screen.

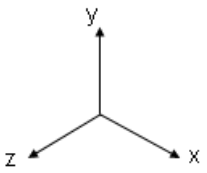
The chart types are listed in the left panel of the wizard. On selecting a particular chart category the various sub types are listed on the right panel. You can select the desired chart type from the different options listed. The different types of chart are listed below:

3D



A 3D chart displays data using three axes, X, Y and Z. The orientation of these axes follows the right-hand rule: make a fist with your right hand, and extend your thumb to point to the right. Extend your index finger so it points upwards, making a 90 degree angle with the thumb. Finally, extend your middle finger so it is at 90 degrees to both your thumb and index finger. You now have the directions of the X (thumb), Y (index finger) and Z (middle finger) axes. These are shown in Figure 4.17, “XYZ Axes”.

Figure 4.17. XYZ Axes



Four kinds of 3D charts are supported, Area, Column, Line and Surface. In each case, the X axis may reference any kind of field, such as strings, numbers etc. The Y axes can only hold numeric values, so the wizard will only let you select from the numeric fields of your datasource. The Z axis can reference any kind of field, but discrete values, ie. strings make most sense. The Z axis can't key the values effectively over a continuous range, such as a floating point number.

The chart preview tab on the Chart Wizard allows you to preview what your chart will look like. In addition, for 3D charts, you can also adjust the camera angle here, to look at the chart from a different orientation. To do this, press and hold the mouse button and drag slowly up and down and left and right. You will see the chart repainted as you drag the camera around.

Note

3D Surface chart shows values in a range and is also available in Dashboard Designer.

Area

An area chart displays series as a set of points connected by a line, with an area filled in below the line. By displaying the sum of plotted values, an area chart also shows the relationship of parts as a whole. Area charts are typically used to compare values over time.

The Stacked Area charts shows related data groups one above another. The Stacked Area charts are used when you want to compare the sum of multiple sets. The Stacked percentage chart shows the area shows each data set as the percentage of the sum of all sets.

Bar

A bar chart displays series as set of horizontal bars that are grouped by category. The categories are arranged horizontally and the values vertically . Bar charts are used to compare value over categories and are ideal for showing the variations in the value of an item over time, or for showing the values of several items at a single point in time.

The Bar chart types include Bar, Bar Chart 3D, Bar Stacked Chart and Bar Stacked Percentage. The 3D Bars are used to display data in 3D format which is convenient for visually comparing the Bars. This is used for making comparisons between groups of data particularly when you want to compare one period of time to another(You can use Bar graph to compare sales per quarter).

From Repertoire 7.6 onwards, sub-category is supported for both Bar and Column charts. Figure 4.18, “Bar Chart without Sub-Category” and Figure 4.19, “Bar Chart with Sub-Category” explains for themselves.

Figure 4.18. Bar Chart without Sub-Category

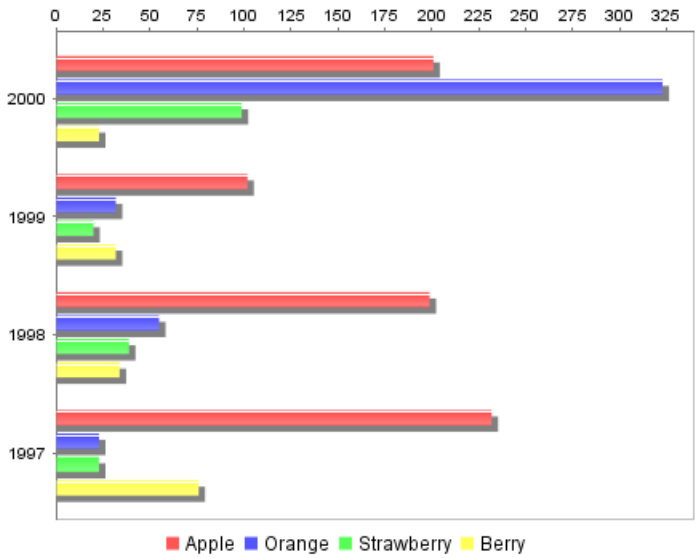
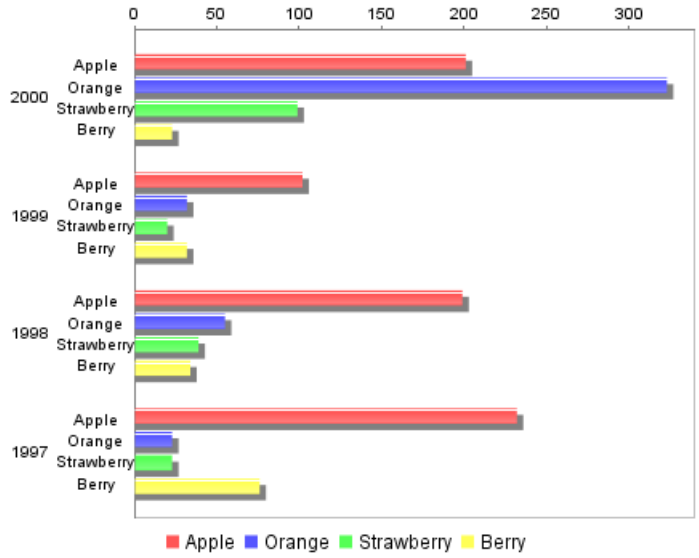


Figure 4.19. Bar Chart with Sub-Category



Line

Line chart are used to display the trends in data at equal interval. Values are represent by the height of the point along the Y-axis and categories are displayed along the X-axis.

This category also includes Wind Chart which has similar properties as the Line Chart.

Column

The Column charts are similar to Bar charts but instead of horizontal bars the data is displayed as vertical bars.

Pie

Pie chart represents single data set as percentages of the whole. The individual slicesbrepresent the category. The size of the slice is determined by the value. These charts are useful for showing percentages. Using the Pie Chart you can display only one set of values for a certain category but if you want to represent multiple values for a certain category the multiple pie charts are used.

This category also includes Ring charts, which have the same properties as a Pie, but have a hole in the middle and do not allow slices to be exploded. Multi-ring charts are also available.

Note

From Repertoire 7.5 onwards, labels can be added to Pie Chart element. This suggests that the Pie Chart will be able to show the respective field names and values. *{0}* will display the field name, *{1}* will display the value and *{2}* will display the percentage of each slice.

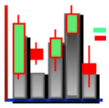
Meter

The Meter chart is used for representing the values in the form of a Meter. Using the meter chart we can show the Overall, Normal, Critical and Warning ranges. Additionally, the Meter chart includes a thermometer type. It is useful for creating charts for management purposes.

Polar

These charts are used to graph according to angles or other criteria particularly suited to a circular format. Polar charts are required in case of financial analysis. These charts can be used to graph repeating data sets that flow into each other. For example you can graph temperatures hour by hour for several consecutive days, representing each day in a different color or pattern.

Stocks



As the name implies, a Stock chart is most often used to illustrate the fluctuation of stock prices. However, this chart may also be used for scientific data. For example, you could use a Stock chart to indicate the fluctuation of daily or annual temperatures.

Candlestick graphs are used when you want to show the stock prices with open, high, low and close values and want to emphasize whether the prices are up or down each day. It can also be used when a similar chart with these four values is to be created. An High-Low-Close must have data sets for high and low values. These charts can also be used to show high and low values similar to Candlestick charts.

Waterfall

Waterfall charts are a special type of Floating Column Charts. A typical waterfall chart shows how an initial value is increased and decreased by a series of intermediate values, leading to a final value.

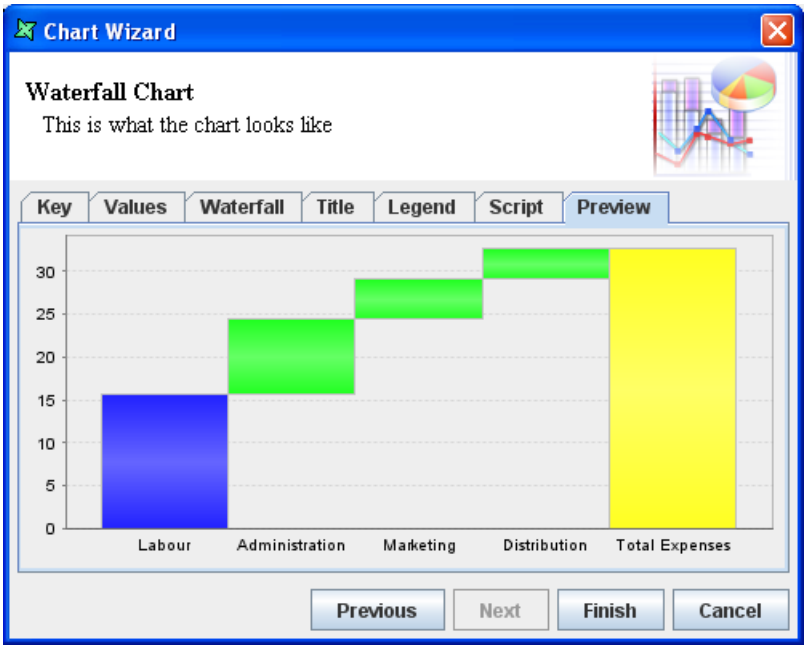
Waterfall charts have very specific data requirements. Here is a sample datasource we will use for illustration:

```
Product     Stage           Value
-----------------------------------
Product 1   Labour          15.76
Product 1   Administration   8.66
```

```
Product 1   Marketing         4.71
Product 1   Distribution      3.51
Product 1   Total Expense    32.64
---------------------------------
```

Every Waterfall datasource requires three fields to be specified, one Key and two Values. The Key in this case is Product. The first value should be the field that describes the composition of the bars, in this case Stage. The second value should be the field that holds the numeric data. In this case Value. If you create a dataset with more or less than two values, it won't render correctly. The output produced for the sample dataset given above is shown in Figure 4.20, “Sample Waterfall Chart”.

Figure 4.20. Sample Waterfall Chart



XY

XY charts are used to graph paired data. This type of chart either shows the relationship among numeric values in several data series or plots two groups of number as one series of xy coordinates. It shows uneven intervals 3/4 or clusters 3/4 of data and is commonly used for scientific data. When you arrange your data, place x values in one row or column, and then enter corresponding y values in the adjacent rows or columns. The scattered graph consists of plotted points "scattered" around an X-Y grid. The pattern may reveal a relationship between the two variables measured by the X and Y axes. This is more or less similar to the Line chart but in this chart type the key and value data should be numeric unlike Line charts.

Scattered charts are used to view actual measurements or observations on a grid, possibly revealing patterns and trends in those data.

Bubble Chart and Heat Map Chart are in this category as well.

Chart Properties

The Area, Bar, Line, XY, Column and Line charts are more or less similar because we can select a single field for the Key but multiple fields as values.

After choosing the chart type and clicking the Next button the screen allows chart-specific properties to be entered. Many of the tabs on these screens are similar. The common tabs seen in all the chart wizards are described below:

Key: The key provides Labels for the Chart elements. For instance in the column chart the key is plotted along x-axis one for each set of columns. In the wizard tab Key can be specified by selecting the corresponding Field Type from the combo box. The Field type can either be a data field or Script.

Values: When a chart is created, you should add at least one value series to the chart. Values determine the size of the chart element for each Key. For example, values determine the height of a column in a column chart and the size of a slice in a pie chart. If you define a single value series, then a single chart element is displayed for each Key. For example, a simple column chart with one value series displays a single column for each Key. If you define multiple values, the chart will display a chart element for each value series.

Pie charts can only support a single set of values whereas many other chart types support multiple values. The GUI of the value tab window is specific to the chart types and will be explained in the example for the corresponding chart types.

Title: The chart title can be specified by selecting a Field type from the combo box. The possible field types are Field, Operation, Script and Literal. Usually the title will be a literal text string, like "Sample Elixir Chart", but you can also use data from your datasource to define the title. The position of the Title can be selected from the combo box. By clicking the Colour or Font button the presentation of the title can be controlled.

Legend: The legend displays the label for each data series in the chart. It is used to identify each data series used in your chart by using the colour, symbol or pattern. If there are multiple data values Report will identify each data series by using a different colour or pattern. The Legend title can be of any field type selected from the combo box. The available field types are Field, Operation, Script and Literal. Just like the title, the position, font and colour of the legend title can be controlled.

Script: JavaScript code can be used to manipulate the Chart element. Elixir charts are built on JFreeChart [http://www.jfree.org/jfreechart/] and the JFreeChart instance is accessible as `this` from the JavaScript to allow further customization. For example, to put a background image on a chart you can use:

```
url = new java.net.URL("file:/C:/Photos/Sample-01.jpg");
im = Packages.javax.imageio.ImageIO.read(url);
plot.backgroundImage = im;
```

Elixir provides a helper class to allow easy changing of the default chart colours. Here's an example:

```
importClass(
 Packages.com.elixirtech.chart2.ui.CustomDrawingSupplier);

paints = ["Pink","Green","Orange","Blue"];

cds = new CustomDrawingSupplier();
cds.setPaintNames(paints);
plot.drawingSupplier = cds;
```

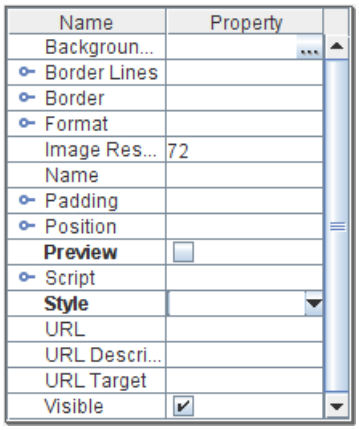
Preview: This displays a chart sample of how it will look like when it is rendered but it is just an indicator as it is displayed at design time it will not have any idea of the runtime data grouping and sorting. Preview is very useful for testing scripts.

Chart Preview on Workspace

Previewing of Chart element on the workspace can be disabled by unchecking the checkbox in the `Properties` panel, as shown in Figure 4.21, “Properties Panel”. With the *Preview* function enabled,

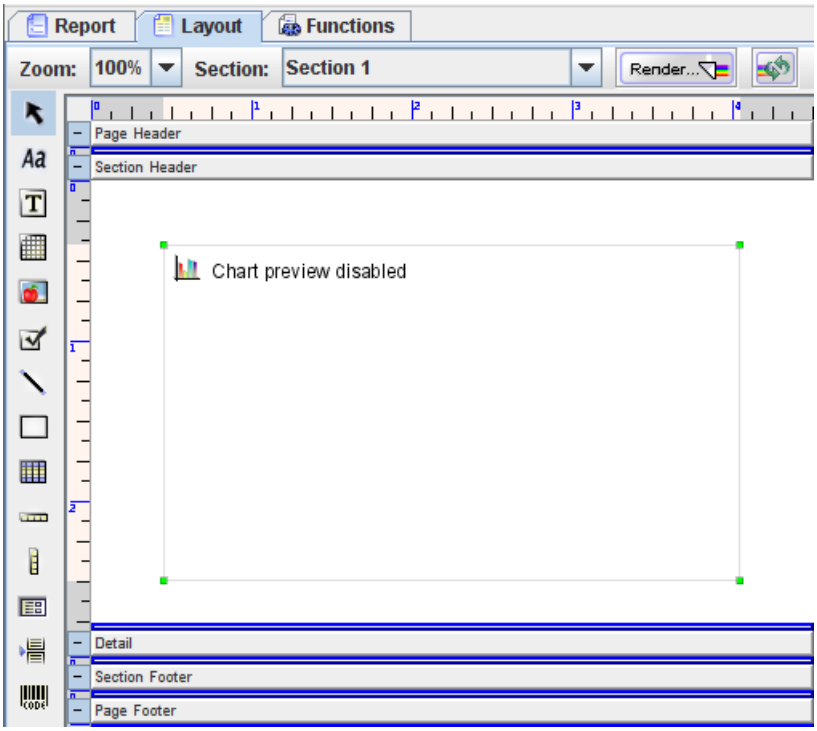
any changes made including the size, colour and position invokes a re-render of preview. Thus, increasing the chances of performance issues.

Figure 4.21. Properties Panel



When the *Preview* function is disabled, the Chart element on the workspace will show that previewing of the Chart is disabled, like in Figure 4.22, “Chart Preview Disabled”.

Figure 4.22. Chart Preview Disabled



Creating Charts

The steps followed for creating Area, Bar, Column, Line, Waterfall and XY Charts are similar except for that in Column Charts, it consists of Column tab and in Area Charts there is an Area tab but the properties in these tabs are all the same. The steps for creating a Column Chart is given below. In this example you can see the difference between the Column Chart by selecting OverAll, OverGroup, etc.

Here's how to add a Column Chart based on the Company sales for four consecutive years.

1. Add a Blank report with ChartData.ds as the default data source which points to the Fruit.csv file.

2. Invoke the Section Wizard on Section 1 and sort based on CompanyName in Ascending order and group on Each Value. Make sure the group header is enabled.

3. Select the Layout tab and place the Chart element in the Section header. In the Wizard that appears select Default from the DataSource combo box. Select the Over All option and click on the Next button.

4. In this screen select the Column category and Column as the chart type. Click on the Next button.

5. In the Key tab select Company from the fields listed in the table.

6. In the Values tab click the Add button to add a value. Select Operation from the Field type combo box and Sum from the Operation combo box. Select 2000 from the list of fields and click on the Ok button. Follow the same procedure and add the 1999, 1998 and 1997 fields.

7. Select the column tab and enter "Years" as the label for X Axis and "Sales(Company)" as the label for Y axis. button.

8. Select the Title tab. By default Literal is the field type. Enter Company Sales(Over All) in the text field and select North as the position. Select Legend tab and enter the legend title as Company and select South from the position combo box.

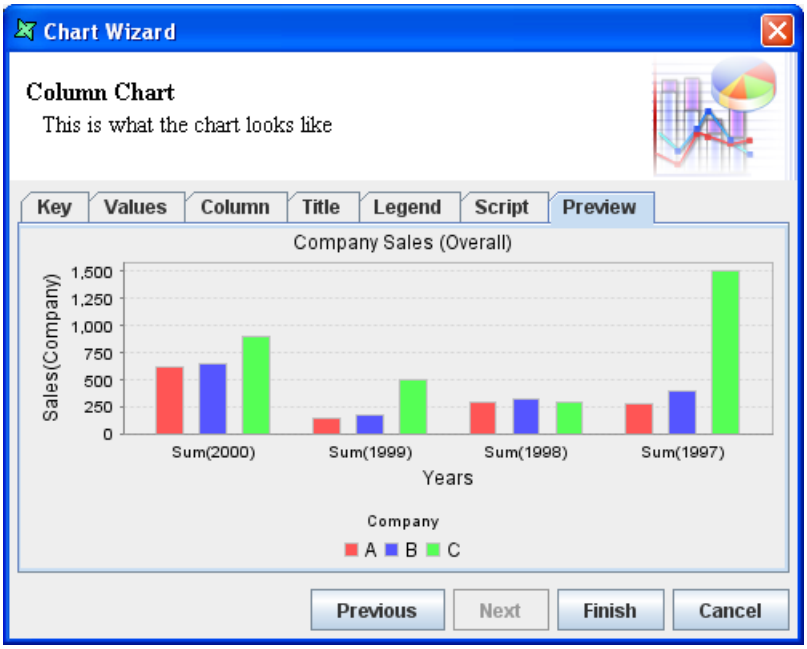
9. After entering all the values the preview should appear as shown in Figure 4.23, “Column Chart”. Click on the Finish button to embed the chart in the report.

10. Next, expand the group header so that it is large enough to accomodate a chart. Copy the Chart element from the Section Header and paste it into the Group Header. You can use the popup menu for copying and pasting, or the accelerator keys Control-C and Control-V. In order to paste into the group header using the keyboard, you must select group header first. The currently selected header shows a highlighted ruler down the left hand side.

11. Now edit the Chart element in the Group Header so that it is Over Group instead of Over All. You can also change the chart title to "Company Sales (Over Group)" so that the charts can be distinguished.

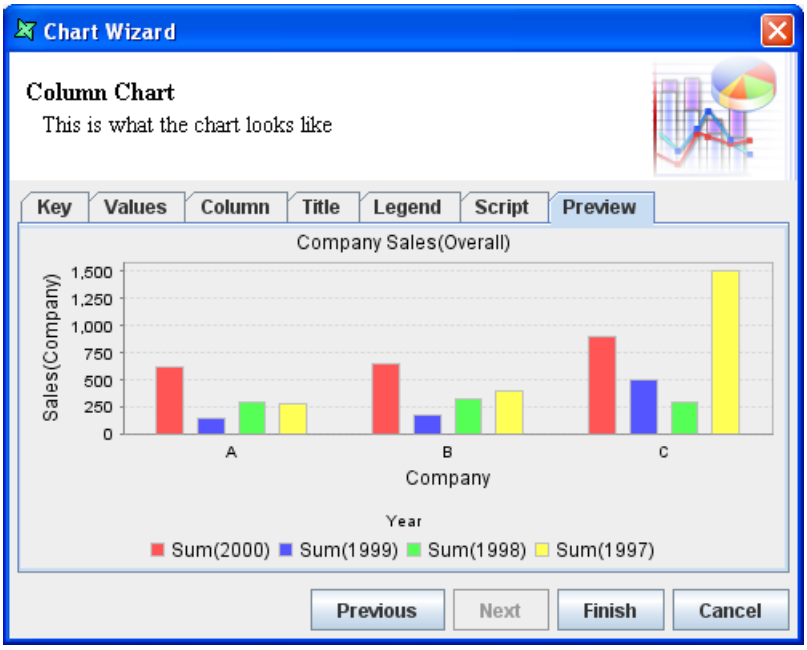
12. Now click the Render button and render as Glint. You will see that the Chart element in the Section header displays the Over All sum(sales) of the companies for the respective years. The Chart element placed in the Group header will display the Over group sum(sales) of each company.

Figure 4.23. Column Chart



In the Column tab an Invert Data check box is available. There are three data items per point in a chart: Key, Value and Amount. So when you turn on the Invert Data check box the Value and Key are switched over. The first data item controls the Overall grouping, including legend and allocation of colours. The second data item controls the clustering of values along X-axis (Column Chart). The third data items gives you the height of the bar. So, by inverting the data we get the preview as shown in Figure 4.24, “Inverted Data Column Chart”.

Figure 4.24. Inverted Data Column Chart



If in the above example Key(CompanyName), Value(Year) and Amount(Sales) are the data items. When the invert data check box is selected the Company Name and Year data items are switched but the Sales remains unchanged. The X-Axis title and Legend title have been modified in this example to reflect the inversion of data.

There are a number of additional chart properties that can be controlled from the wizard. These concern the rendering of values on the columns (or other chart types) and are described below:

Position: This attribute controls the position of the plot value label of each data item on the chart. The various options that are available are Center, Inside 1 to 12, Outside 1 to 12, etc. They are similar to the position of the hands in the clock. If for instance you are plotting negative values then Outside6 would be the position for the negative column.

Anchor: This is the enumeration of position which a plot value label can take. That is once you know the point location of the value you need to map to a point on text. If for instance in case of column you may want positive column values to have their center baseline at the point, but for positive bar values, you may want left half-ascent. The available values are baseline, half ascent,etc.

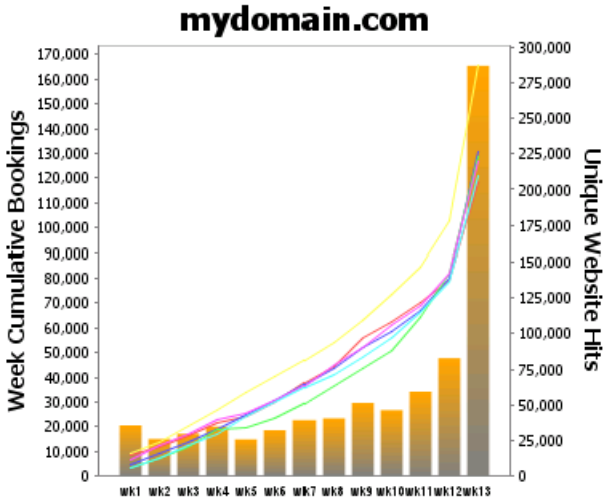
Pivot: This attribute specifies the point of rotation about which the plot value label has to be rotated. The available values are Baseline left, Half Ascent, etc. If you want to rotate a label you have to select a pivot value about which it has to be rotated and specify the angle of rotation.

Angle: The angle in degrees by which the Plot label value should be rotated (about the pivot point).

Creating Dual Axis Charts

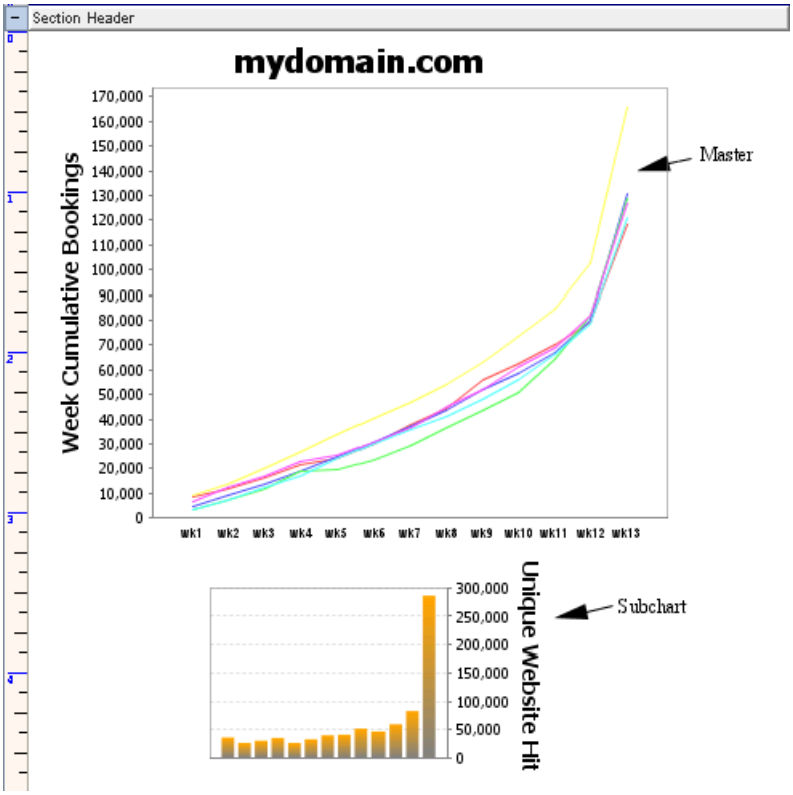
In order to create a dual axis chart like the one shown in Figure 4.25, “Dual Axis Chart”, you need to be able to control each side of the chart independently. To make this easy, you can create two independent chart plots and then merge them into one chart.

Figure 4.25. Dual Axis Chart



We will call the first chart, that will show the final result the Master. The other chart, which will be merged in to the Master we will call the Subchart. The two charts to be merged are shown in Figure 4.26, “Dual Axis Chart Design”.

Figure 4.26. Dual Axis Chart Design



You will notice that one has the Y axis on the left and the other on the right. To put a Y axis on the right you need to add this script:

```
importClass(Packages.org.jfree.chart.axis.AxisLocation);
plot.setRangeAxisLocation(AxisLocation.TOP_OR_RIGHT);
```

In the designer you will see separate charts, so you can configure them independently. In the rendered output you will only see one combined chart.

1. Create the two charts and ensure the Subchart Name is set ("Subchart" is used here).
2. In the Properties panel, set the Subchart visibility to false.
3. Go to the Shapes tab, right click on Master and select Wizard to access the Chart Wizard dialog box.
4. Click Next until you reach the Line Chart section. Go to the Script tab.
5. Key in the following code, modifying it based on your design:

```
importClass(Packages.com.elixirtech.report2.logical.renderer.\
  ChartRenderer);

if (cxt!=null) // real rendering
{
  var subChart = cxt.getRawElementByName("Subchart");
  var subPlot = ChartRenderer.buildChart(cxt,subChart).plot;
  var axis = subPlot.getRangeAxis();
  axis.setAutoRange(false);
  plot.setRangeAxis(1,axis);
  plot.setDataset(1,subPlot.getDataset());
```

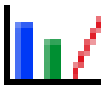
```
  plot.setRenderer(1,subPlot.getRenderer());
  plot.mapDatasetToRangeAxis(1,1);
}
```

(the text here is wrapped at the \ character on the first line - remove this and concatenate the first two lines together).

Note

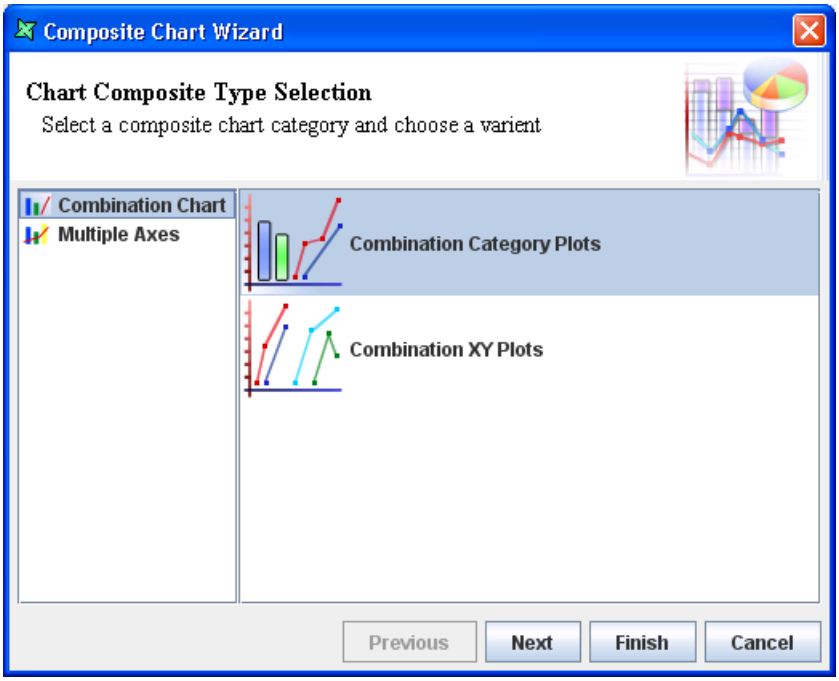
When you have finished designing your two charts, you can put the Subchart anywhere you want - even on top of, or behind the Master as it won't appear in the output. This way it doesn't occupy space you need for other report components

Composite Chart



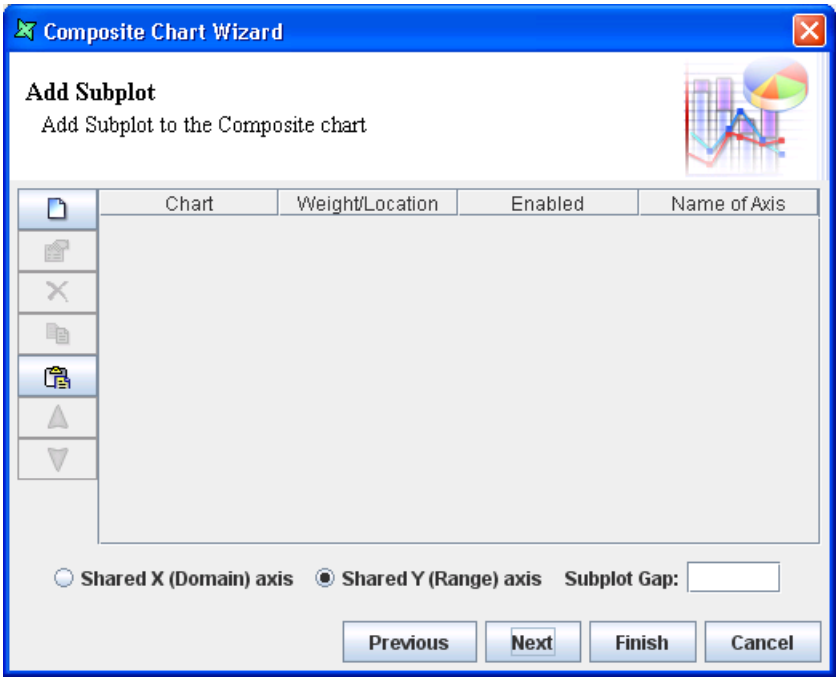
Composite Chart is an advanced version of the Chart element. It allows multiple charts to be displayed in a single chart. Composite Chart creates a combination chart or chart with multiple axes. Area, Column and Line Charts can be combined.

Figure 4.27. Composite Chart Type



Adding a subplot adds an additional chart into the Composite Chart. The creation of each subplot is similar to the process of creating a convention Chart element. The only difference is that the process is more simplified with fewer chart types. In the section called “Case Study 1 - Composite Chart” is a sample of creating a simple Composite Chart element.

Figure 4.28. Add Subplot



Note

Retain Key Order is available in Composite Chart.

Combination Chart

Combination Category Plots allows the display of 2 or more Area, Column and Line Charts together side by side. Multiple XY Charts can be achieved by selecting `Combination XY Plots`.

The figure to be entered for `Weight/Location` determines the proportion of the chart. `Subplot Gap` determines the amount of spacing in between each chart. The value is in terms of pixels.

Multiple Axes

Multiple Axes displays 2 or more Area, Column and Line Charts together by overlapping each other while sharing either the X-axis or the Y-axis. When one axis is shared, the other axis will have additional axis depending on the number of charts in the Composite Chart. When the X-axis is shared, the location will be either right or left. If the Y-axis is shared, the location will be either top or bottom.

RTF

Rich Text Format is a file format developed by Microsoft that allows you to define text files with formatting, font information, text color, etc. The RTF renderer will read the text from a control source, which might be a Field, Literal, Script or URL. This means RTF text can be read from your datasource, or from a file on disk. RTF is the preferred way to obtain attributed text, where the font, colour, alignment, even line-spacing can vary within a block of text.

To use the RTF component as a literal, you need to paste the raw text of an RTF file into the RTF Wizard (or type it directly, if you know RTF syntax). For example, you can create a simple text with

WordPad (on Windows) and save as RTF. Then open it with NotePad, to see the actual RTF codes - lots of { and }. This is the contents you need to paste into the RTF Wizard for a literal value. Alternatively, this is the data you need in the database, if you want to read it as an RTF Field.

Cube Table

The Cube Table element allows two-dimensional cubes to be embedded in a report. Each cube has two dimensions, one representing the table rows and one the columns. Each dimension may consist of multiple levels, defined by data fields. The records are grouped by these levels and those records forming the intersection of the row and column levels are rendered using one or more cube measures, for example Sum or Average.

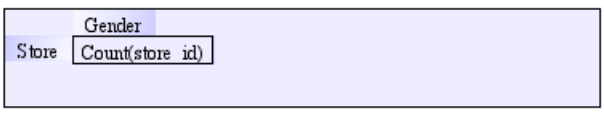
An example Cube Table output is shown in Figure 4.29, “Sample Cube Table”. There are a lot of options to control the formatting and presentation of the data.

Figure 4.29. Sample Cube Table

				Gender		
				F	M	
				Total	Total	Total
Store	USA	CA	Beverly Hills	3426	3389	6815
			Los Angeles	3864	4343	8207
			San Diego	4048	4047	8095
			San Francisco	653	672	1325
				11991	12451	24442
		OR	Portland	3957	4307	8264
			Salem	6580	6767	13347
				10537	11074	21611
		WA	Bellingham	676	704	1380
			Bremerton	3753	4123	7876
			Seattle	4301	3655	7956
			Spokane	3797	3600	7397
			Tacoma	5538	5646	11184
			Walla Walla	641	698	1339
			Yakima	1597	2055	3652
				20303	20481	40784
				42831	44006	86837
				42831	44006	86837

The designer view of the cube table shown above is illustrated in Figure 4.30, “Cube Table Designer View”. Within the cube table you can see three additional subcomponents - the Row Header (called Store) the Column Header (called Gender) and the Measure Count(store_id). Any additional measures will also appear here.

Figure 4.30. Cube Table Designer View

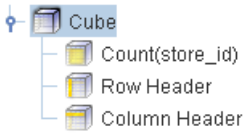


To create a Cube Table, click on the icon and drag a rectangle at the desired location on the report. A wizard will appear that allows you to choose the datasource from which the table will be built. You

can choose the range of records to be used in formulating the report, either Over All, Over Group (useful if the Cube Table appears inside a Group Header or Footer), or Over Record. Usually you will choose Over All or Over Group. The next step is to click `"Open Cube Wizard..."`.

The Cube Wizard is described in the Elixir Data Designer documentation (See Elixir Data Designer.pdf, Chapter 4). A sequence of pages takes you through the definition of hierarchies, dimensions and measures for the cube, based on the chosen DataSource. Clicking Finish will add the Cube Table to the Report and it will appear in the Shape tree as shown in Figure 4.31, “Shape Tree View”. Each dimension and measure in the Cube Table has distinct properties, as does the Cube Table as a whole.

Figure 4.31. Shape Tree View



Cube Table Properties

The Cube Table can grow wide when there are lots of elements in the column dimension. Two additional properties are available to control how the table is sized. "Width Can Grow" and "Width Can Shrink" indicate whether the table can be resized to best fit the horizontal size of the contents. You will notice in the example above that the rendered Cube Table is narrower than the design mode version. This is because the cube has Width Can Shrink set to true. Note than when Width Can Grow is set to true, it will still be clipped to the number of horizontal pages you have chosen in your page setup.

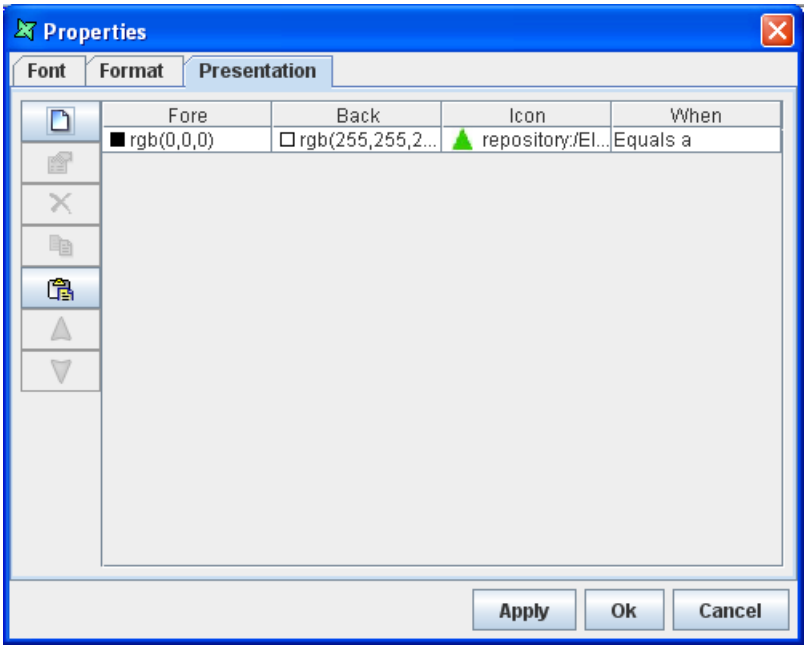
Header Properties

Each header defines two background colours, "Top Left" and "Bottom Right". Because each header may contain multiple levels, each level is represented by a step in the gradient of colours from TopLeft to BottomRight. If you choose the same colour for both, then every level will have the same colour. Note you can vary colours by transparency, in which case each level will gradually fade to white (or whatever colour you set the Cube Table background to). Border lines can also be turned on or off on different sides to highlight the levels in the tree.

Measure Properties

Each measure has properties similar to a Field: alignment, padding, format etc. In addition, each measure has a Properties panel to control presentation. A sample presentation is shown in Figure 4.32, “Measure Presentation Options”.

Figure 4.32. Measure Presentation Options



The presentation options consist of a list of tests, which might be content tests, eg. Value > 1000 or may be structure tests, eg. RowLevel==3. Each test is associated with a foreground and background colour, and an optional icon. When formatting a value for display, the list of conditions is scanned from top to bottom looking for a match. When a test returns true, then the associated foreground, background and image are used to render that value. Once a match is found, subsequent tests are ignored.

The most flexible presentation test is through JavaScript. Several variables are predefined so that you can use them in your tests.

- **Col**: The current column (zero-based).
- **ColLevel**: The current column level, zero for normal columns, one for totals, two for next level totals etc.
- **Row**: The current row (zero-based).
- **RowLevel**: The current row level, zero for normal rows, one for totals, two for next level totals etc.
- **Value**: The value of the measure.

The sample shown above sets the background to white for all "normal" rows (i.e. not totals) as shown in the cube table at the beginning of this section. Combinations of these conditions allow for very flexible value rendering. For example, RowLevel==0 && Value>100 will test true for those normal elements whose value is greater than 100, but won't include totals (which are usually bigger).

Callback

A Callback element is a component plugged in to Elixir Report Designer to allow specialized rendering. Java code can be plugged into the render engine to handle specific data formats. As an example, Elixir

Report Designer ships with HTML and RTF. Others may be available or already installed in your configuration. We will focus on HTML here, because that is available in all installations.

Select and place the Callback element in the report layout. The Callback Wizard appears listing the component types that are available.

When you select the HTML and click on the Next button the screen allows the specification of the control source - it might be HTML read from a file (or URL), literal HTML, or HTML from a datasource. If you select Literal as the Control Source type and enter the following script:

```
<h1>This is a test report</h1>
```

Then on rendering the report, the above sentence is displayed as a Level 1 heading. Similarly all HTML 3.2 tags are interpreted into text and displayed when the report is rendered. The HTML render component uses Sun's HTML rendering technology, so support is limited to the features Sun's renderer supports.

Report Parts

In this section we will take a look at the individual properties of the report parts.

Resizing report parts

The height of the report parts can be adjusted by using the drag bar. Select the report part that has to be resized and position the cursor near the drag bar. The default cursor changes to a hand cursor. Now by pressing the mouse down drag bar up or down till the required height of the report part has been set.

The height of the report parts can be altered according to the requirements. Position the cursor in the report part section you wish to move or resize. It should be moved towards its drag bar. The mouse cursor turns to hand shape. Now, select the drag bar with the cursor and pull it down. The height of the report part will be altered. Alternatively, to change the height of the report part it will be necessary to perform the above mentioned function.

Table of Contents (TOC)

The properties wizard of the Report parts can be invoked by either double clicking on the background or from the popup menu. The wizard consists of the TOC, Format and Scripts tabbed panes. The Format and Scripts tabs have already been described in previous sections, so we will focus here on TOC which is an acronym for Table Of Contents.

The TOC tab contains the fields similar to that of the Field tab in the Data Field and other elements. Additionally, there is a TOC Enabled checkbox which when turned on allows enabling the TOC feature at that level.

The TOC properties can also be accessed from the property table of the report part. The TOC node in the properties table includes the Data, Enabled, Format and Locale. The Data value can be entered directly in the text field or by clicking the button in that field the Properties Wizard is invoked from which the data source field can be selected.

The Format value can be entered directly in the field of the property table. Alternatively, by clicking the button in the field the Properties Wizard is invoked in which the Format properties can be set. The table of contents can be enabled by turning on the Enabled check box which is similar to the TOC Enabled check box in the TOC tab. The Locale property is discussed earlier in this chapter.

Note

Table of Contents produces a tree of bookmarks and is currently only supported in the PDF output format.

Properties

The Section header, Detail, Section Footer, etc have the following properties in addition to the above mentioned properties.

Force New Page: The Default value of this property is None. If the default value is specified then a new page will not be added. The other possible values are Before, After, Before and After. If these values are selected then the data in the specific report sections are printed on a new page at the respective location.

Vertical Align:

There are three possible values: Top (the default), Center and Bottom. **Top** positions the report part in the normal flow of the report, so most of the time, this is what you want. **Bottom** positions the report part at the bottom of the current page, just above any page footer. Subsequent parts therefore appear on the next page. **Center** positions the report part in the center of the remaining space on the page. Subsequent report parts will render immediately below, if there is space. Center can always be used in conjunction with Force New Page = After, to ensure subsequent parts are pushed to the next page.

Bottom Padding:

This Bottom Padding property describes how much space to be included between the bottom border and the data. This property is useful when the CanShrink property check box is selected otherwise the end of the records will touch the bottom of the component. In this case as a value is set for the Bottom Padding property some amount of space will be available between the component and the records.

Caption:

The Caption property is similar to the Name property of the report elements. This property is useful when you want to display the caption for the grouped data for instance in case of Elixir Report Mobile Edition(ERME) when you want output data to small screens like hand phones.

Case Study 1 - Composite Chart

This case study uses *FruitSales.ds* to create a simple example to demostrate the use of Composite Chart.

1. Create a new blank report, with any name, for example, *sample* and use *FruitSales.ds* as the datasource.
2. Click on the icon for Composite Chart, and drag a space in the Section Header.
3. Select `Multiple Axes` then `Category Plot with Multiple Axes`.
4. On clicking `Next` is the wizard page to add the subplot.
5. Click on the Add icon, Chart Wizard opens. Select `Over All`, then click on `Next`.
6. Select `Column` then `Column 3D`. Click `Next`.
7. For `Key`, select *Fruit*. For `Values`, add *1998* and *1999*.
8. Click `Finish`.

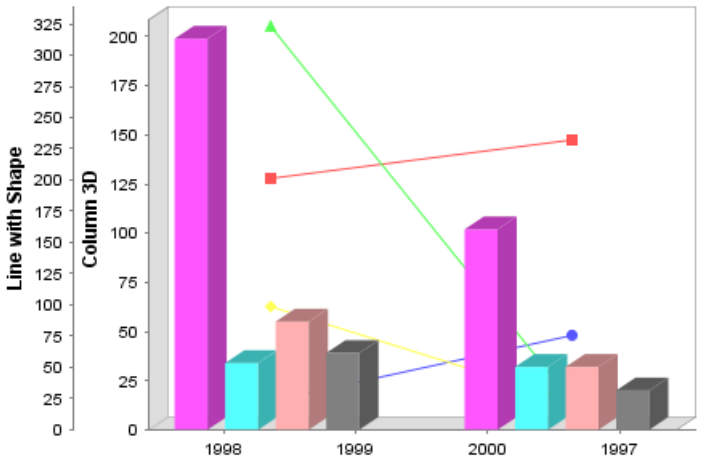
9. Follow step 5 to 8. Except when choosing the Chart type, select `Line`, then `Line with Shape` instead of `Column` and add *1997* and *2000* for Values.

10. At the bottom of the wizard, check `Shared X (Range) Axis`.

11. Enter the name of the chart in `Name of Axis` for easy identification when viewing the chart.

12. Click `Finish` to finalize all the editing. The Composite Chart will appear like in Figure 4.33, “Composite Chart”.

Figure 4.33. Composite Chart



Note

If the Composite Chart has a shared X axis and each of the charts has a name for their respective axes, the name entered in `Axes` tab, `X Axis` field will be overwritten. On the other hand, if nothing is entered for the respective axes, the name in `X Axis` will be the name for all the X axes. This is applicable to Y axis as well.

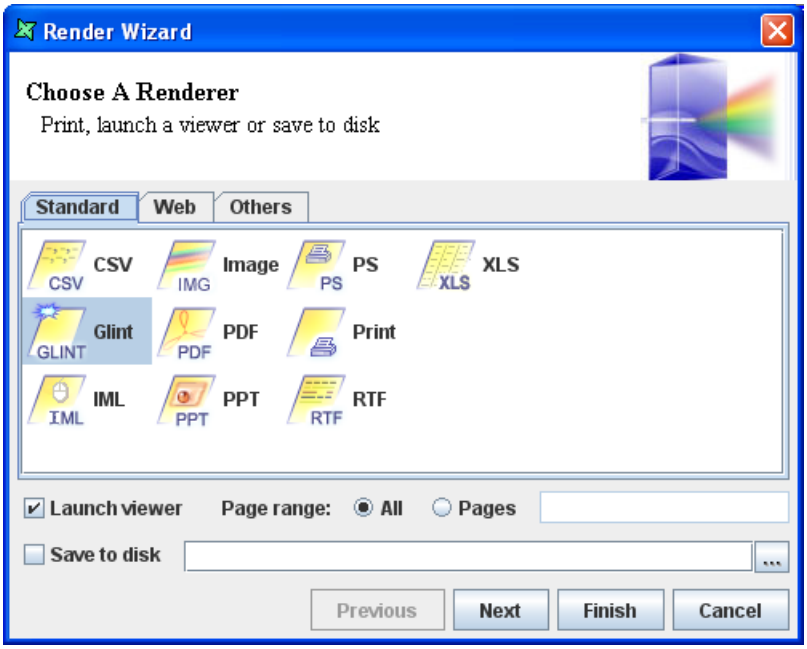
Chapter 5
Report Rendering and Output Formats

Report Output formats

Elixir Report Designer supports a wide range of output formats, including vector-based formats such as Glint and PDF, bitmap formats, including PNG and JPEG as well as structured text formats such as XML, CSV and HTML.

Rendering is controlled by the Render Wizard, which can be invoked from the Layout tab of the Designer or from the popup menu of a report template in the Repository. The wizard appears as shown in Figure 5.1, “Render Wizard” Start by selecting the desired output format from the first screen. By default the Launch Viewer check box is selected. You may have to configure the appropriate viewer for certain output types by editing config/mailcap.txt. To save the file to the disk select the Save to Disk check box and enter the save location in the text field. Alternatively, by clicking the button on the right of the text field the location can be selected from a Save dialog.

Figure 5.1. Render Wizard



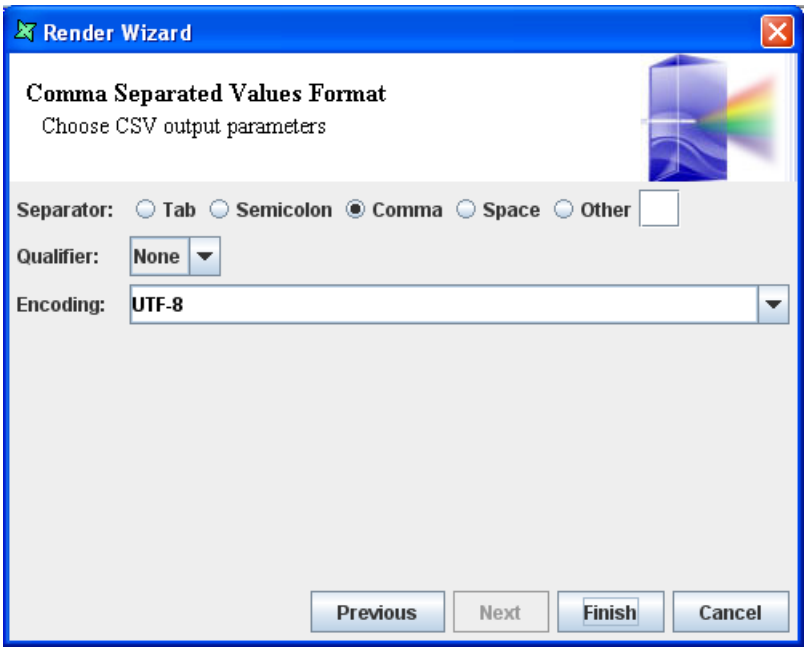
Each output type has different parameters that can control the rendering process. Clicking Next will take you to a different page depending on your choice of output type. Any values you set for a particular output type will be stored and recalled each time you re-render the same report. If you have set the values, or are happy with the defaults, just click Finish on the first page, there's no need to advance to the next.

CSV

The CSV Renderer is used to generate a text based(.txt) or CSV(.csv) file. The CSV file that is generated allows direct import into a spreadsheet such as Excel or OpenOffice.

The CSV render properties are shown in Figure 5.2, “CSV Options”. Select the separator from among the types shown or if you wish a custom separator choose Other and enter the separator in the text field.

Figure 5.2. CSV Options



Glint

Glint is Elixir's native vector graphics format. It is designed to be extremely compact to minimize network traffic and reduce memory requirements on low-end machines, including Java2 Micro Edition devices. It supports precise twip-based vector graphics and page-on-demand access. The Glint files are saved with .glint extension.

The Glint render properties contain a single Tree checkbox. By default Glint does not render in Tree mode; this is primarily for use with Elixir Report Mobile Edition(ERME) when we want to output data to small screens like handphones. Instead of producing long documents it breaks up the documents into multiple hyperlinked pages in a tree structure.

In order to generate the Glint file in tree format you need to set the Table of Contents to true by selecting the enabled check box in the property sheet for the Group Header you want to use. All child details will be rendered on subsequent pages with hyperlinks to them from the group summary. The Data to be displayed in the hyperlink is entered using the standard Data Wizard.

HTML

HTML is an acronym for HyperText Markup Language, a legacy syntax used to structure and link text and multimedia documents which is still used extensively on the World Wide Web.

Two HTML outputs are supported, this HTML is a single simple HTML document, with no supporting files. See the section called “HTML-Zip” for a more comprehensive output, that can include multiple files, with navigation and embedded images.

The HTML render properties consist of a single option:

- **Skip Header**: The skip headers check box when selected will remove the HTML headers from the output and supply only the contents of the body. This makes it easier to embed the output into other documents.

Because this renderer is targetted at a cut-down, simple HTML output, for ease of integration with other web interfaces, it only generates a single file. This means images visible within the designer won't appear in the output, unless they are identified by external URLs (for example http or ftp protocols). Also, images generated by the render engine, such as charts and RTF components, won't appear because they have no external URL to identify them. The images themselves behave differently as well. HTML has no concept of image clipping, so choosing the `Clip` SizeMode will produce a normal unscaled image, whereas both Zoom and Stretch will produce a stretched image.

If you want to see all of the components exactly as they appear in the designer, perhaps spanning multiple pages and with navigation, then you need to use HTML-Zip, which preserves all of the report information, supporting images and other files within the zip.

HTML-Zip

The HTML-Zip render properties consist of several options:

- **Streamed**: The report will be rendered as a single (possibly long) HTML file. No page headers or footers will be rendered.
- **Paged**: The report will be rendered to span across multiple HTML files, with each page showing a page header and page footer.
- **Tree**: The report will be broken up into a tree structure, based on the Table of Contents (TOC) field in group headers and details. The first level will show the root, along with the names of the groups (defined by TOC again). Clicking on a groups will drill down to the next layer in the tree.

  The Highlight Background and Normal Background fields allow you to choose colours that should be used used to create hover effects as the mouse moves over a clickable group, to highlight that an action can be performed. Any colour value that is understood by HTML can be used here.
- **SVG Pass-Through**: The SVG Pass-Through check box when selected will not convert SVG graphics to images, but instead embed the SVG into HTML. This results in smaller, higher-quality output if your browser supports SVG. SVG plugins are available for most popular browsers.
- **Skip Headers**: This option removes all HTML headers so that the rendered content can more easily be inserted into an existing portal or framework.

The HTML files are saved to disk as a zip file containing all supporting images and files. For a simpler, but more restricted HTML output, see the section called “HTML”.

Image

The image render allows rendering into JPEG, PNG, BMP and WBMP image files. JPEG sacrifices some image quality to achieve very high compression, whereas PNG supports lossless images with millions of colors and produces background transparency without jagged edges. It features compression, transparency, and progressive loading, like GIF, but it is free of patent restrictions. BMP does not lose any digital information when they are altered and saved. As for WBMP, it is a graphic format optimized for mobile computing devices.

This renderer outputs in Paged mode and provides a separate image file for each page of the report. The set of image files is zipped.

The Image render properties contain a single Tree option. See the description of Tree in the section called “Glint” for more information.

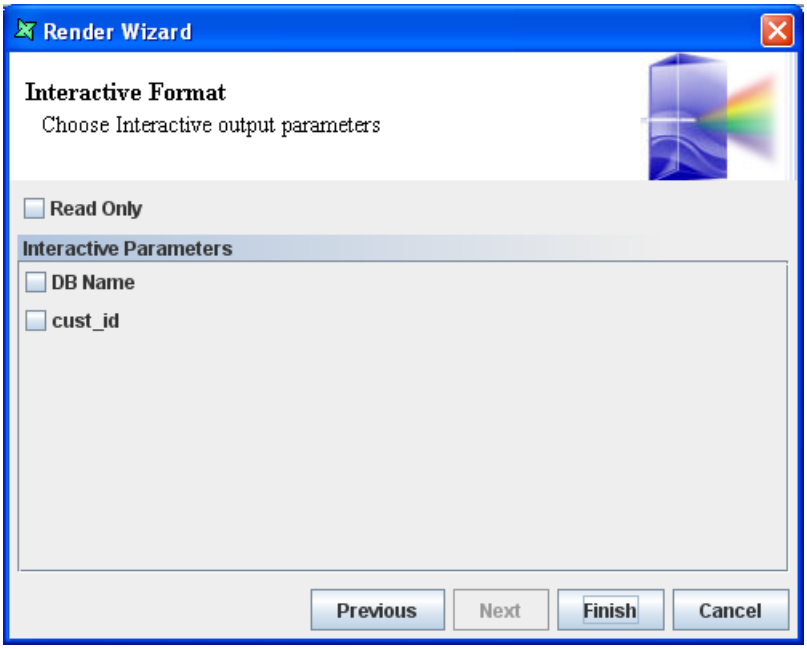
IML

Interactive Markup Language (IML) is a format that allows the generation of a report+data bundle that is rendered interactively in the Elixir Report Interactive (ERI) tool. Using this tool you can modify rendering parameters and even the data to generate fresh reports, without requiring any database connection.

IML works by embedding both the template and data values within the IML file. All data files required will be automatically determined and the records saved. If parameters are used to extract the data, then these are supplied at rendering time, and only this data will be saved. The exception is the Composite datasource, where processing is not done at this stage. Instead, the inputs to the Composite are stored and the Composite is executed as part of the ERI process. This means Composite parameters and Report parameters can still be edited by the user within ERI. It makes no sense to edit the parameters of primitive datasources, like JDBC within ERI because the data was already extracted during the IML generation stage.

The IML render properties are shown in Figure 5.3, “IML Options”. These properties allow you to control whether the IML file is read only. If this option is chosen, the ERI program will not allow the data to be modified. The ERI user will still have a choice of render parameters for on screen viewing, and the option to render the output into any other output format.

Figure 5.3. IML Options



The IML render properties also allow you to choose which report parameters you wish to expose to the user of the IML file. As described previously, there is no point exposing values passed back to primitive datasources because they will have no effect. You should use the checkboxes to enable those parameters which the user can utilise to vary the render output - those that are used by Report scripts or Composites, for example.

LPT

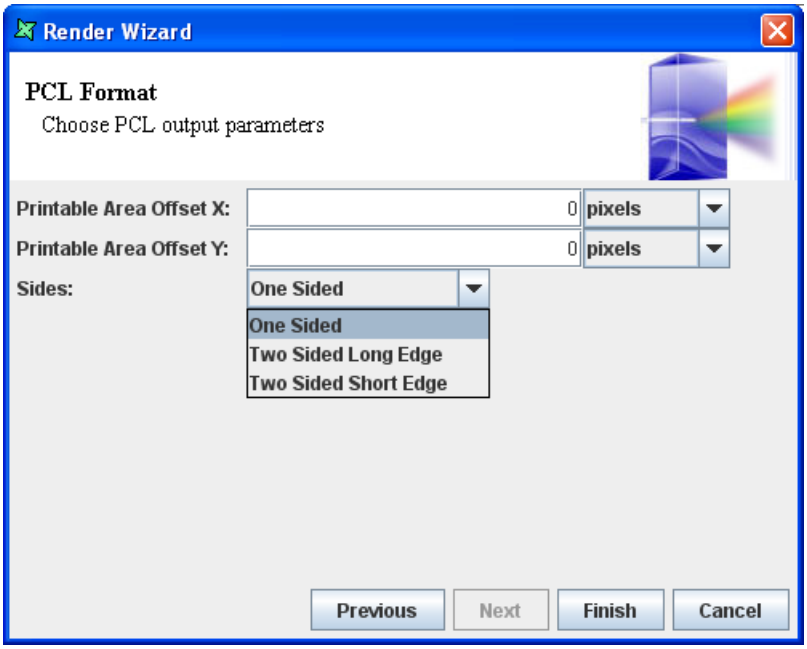
Line Print Text is a text format which positions the text components of the report to approximate the same results on a fixed-width display or line printer.

The LPT render properties consist of two values, Character width and Character height. The values describe the size of a single character on the target device, hence how the layout of the template is positioned. For example the character height indicates how many newline characters are needed to descend the appropriate distance down the page and the character width then indicates how many spaces are inserted to push the text to the correct position. For example, a line of text at 600,600 (twips) would be at location (4,2) on a character grid with the default character size of (150,300) twips.

PCL

PCL is HP's Printer Control Language. Support for this rendering type is limited to a few elements only and is intended primarily for black and white label printers. There are 3 options for `Sides` as seen in Figure 5.4, “PCL Wizard”. *Two Sided Long Edge* is usually used for portrait pages, while *Two Sided Short Edge* is typically used for landscape pages.

Figure 5.4. PCL Wizard



If you intend to use this render type, please verify with some simple renders that the components you intend to use are supported by this renderer. Please contact Elixir if you have additional requirements that are not met by this current renderer.

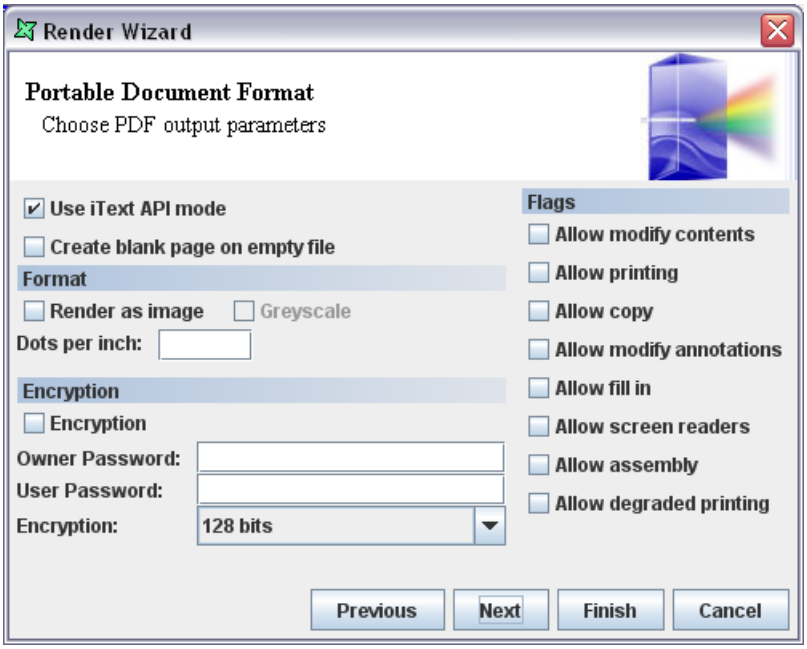
PDF

PDF is a file format created by Adobe, initially to provide a standard form for storing and editing printed publishable documents. Because documents in .pdf format can easily be seen and printed by users on a variety of computer and platform types, they are very common on the World Wide Web.

The renderer currently generate PDF files that comply with the Adobe PDF 1.4 specification. As the PDF format is backwards compatible, this means that the files can be accessed using Adobe Acrobat version 5.0 or later and other PDF viewers and utilities that accept PDF 1.4 or later.

The PDF render properties are shown in figure Figure 5.5, “PDF Options”.

Figure 5.5. PDF Options



Use iText API mode

This flag enables use of the iText API mode which addresses an issue with JDK 1.6 output. By default JDK 1.6 outputs text as glyphs which results in higher accuracy of output, but much bigger files. The accuracy is due to the conversion to vector graphics within the engine, instead of within the PDF viewer. A benefit of the default JDK 1.6 behaviour is that no fonts need to be embedded.

However, where reports need to be transmitted across the network or archived, the increase in file size (which may be a factor of ten or more) can be excessive. By enabling API mode, the renderer outputs characters instead of glyphs to produce smaller files. This may cause some minor discrepancies in the report, for example a line of text may be fractionally longer or shorter because it is now the PDF viewer that chooses the best font match. In API mode, fonts can be embedded to ensure the viewer chooses an appropriate font.

Create blank page on empty file

Create blank page on empty file option allows empty file such as 0 bytes size file to be rendered into the pdf output successfully with blank page created. However, if the option is disabled, 0 bytes file size will fail to render into pdf output.

Format

Image: When this option is selected then in the rendered PDF document each page will be a page-sized image. This results in bigger files, but requires no embedded fonts.

Grey Scale: When this is selected then the output will be displayed in shades of grey instead of colour which will make the file size smaller. If you will only be printing in black and white then it might be useful to reduce memory usage.

Dots per inch: Dots per inch indicates the desired resolution of the rendered image - larger values mean increased sharpness, but bigger file size.

Encryption

When this check box is selected the PDF file is encrypted with user and owner passwords. Encryption allows owners to control certain functions such as printing the file, copying and pasting from the file.

So only when this check box is selected we can set the flag values. If the owner password is left blank, the system will generate a random password - effectively preventing anyone from changing the access options.

The encryption strength either 40 bits or 128 bits is selected from the combo box. The 40-bit strength is the standard one. You can increase a document's cryptographic strength by using 128-bit security.

Flags

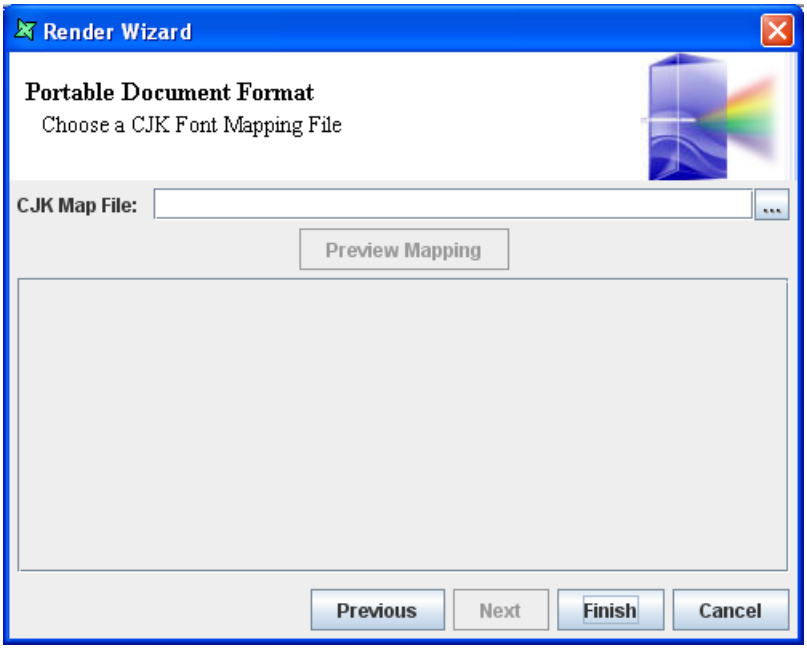
With encryption enabled, a PDF viewer will determine the operations that can be performed on the document based on the list of flags.

CJK Font Mapping

The PDF specification defines 14 built-in fonts. If you use fonts that are not part of the specification they can be embedded in the report (subject to font licensing) or the glyphs can be rendered as vectors, removing the need for the font, but making selection impossible.

CJK Fonts can be very large, so embedding the glyphs or even the font itself results in large files. The next page of the PDF Render Wizard allows font names to be substituted, retaining small files, as long as the CJK fonts are available on the reader devices. The wizard page is shown in Figure 5.6, “PDF CJK Mapping”.

Figure 5.6. PDF CJK Mapping



The CJK Mapping File is a text file, which contains a substitution on each line. For example:

```
Serif=STSong-Light
SansSerif=MSung-Light
```

You can create this mapping file as a text file in the repository, so that many reports can share it and so you can easily deploy it to the Repertoire Server if necessary. Once the mapping file is chosen, all text in the report that use Serif font will be written to PDF as STSong-Light.

Note

The font metrics used in placement of the text will still use Serif, so it is important to choose substitutions with closely matching font metrics. Adobe does not allow the free use of their CJK fonts which prevents use of the actual CJK font metrics.

Common CJK font encodings are defined in /config/EREngine-config.xml. You can add extra definitions here if you need additional fonts, but will need to restart the tool for them to be read.

- STSong-Light: encoding=UniGB-UCS2-H
- MHei-Medium: encoding="UniCNS-UCS2-H
- MSung-Light: encoding="UniCNS-UCS2-H
- HeiseiKakuGo-W5: encoding="UniJIS-UCS2-H
- HeiseiMin-W3: encoding="UniJIS-UCS2-H
- HYGoThic-Medium: encoding="UniKS-UCS2-H
- HYSMyeongJo-Medium: encoding="UniKS-UCS2-H

Font Mapping

Physical fonts need to be installed in locations known to Java Runtime Environment.

Users can add the physical fonts that use a supported font technology by installing them either in the jre/lib/fonts directory within the J2RE, or by installing them in a way supported by the host operating system (copying them into the Fonts folder of windows, using the pkgadd command in solaris, etc).

After you have installed the Windows true type font and PostScript type 1 fonts onto your JVM, you will have to do the font mapping in Elixir Report Designer to load the right fonts for report generation.

When you start the Report the Font types will be loaded along with it.

In the config folder of Repertoire, there is a "EREngine-config.xml" file which controls font mapping. The Java system property "elixirtech.fonts.path" is assigned the value of the absolute path of the font directory.

Table 5.1. Java AWT font types on Window platform.

Java Font AWT Family	font.properties-mapped in JDK 1.4	PDF Type1 Font Family mapped
Dialog	Arial	Helvetica Type1 - Actual font : Arial MT, true type
DialogInput	Courier	Courier Type 1 - Actual font : Courier, Type 1
Monospaced	Courier	Courier Type 1 - Actual font : Courier, Type 1
SansSerif	Arial	Helvetica Type1 - Actual font : Arial MT, true type
Serif	Times New Roman	Times Type 1 - Actual font : TimeNewRomanPSMT,truetype

Table 5.2. Type1 PDF fonts with encoding Cp1252

Window Font Face Name	PDF Type1 Font Name
Courier New	Courier
Courier New Bold	Courier-Bold
Courier New Italic	Courier-Oblique
Courier New Bold Italic	Courier-BoldOblique
Times New Roman	Times-Roman
Times New Roman Bold	Times-Bold
Times New Roman Italic	Times-Italic
Times New Roman Bold Italic	Times-BoldItalic

Table 5.3. List of PDF Type 1 font names

PDF Type 1 Font Name (14)
Courier
Courier-Bold
Courier-Oblique
Courier-BoldOblique
Times-Roman
Times-Bold
Times-Italic
Times-BoldItalic
Helvetica
Helvetica-Bold
Helvetica-Oblique
Helvetica-BoldOblique
Symbol
ZapfDingbats

Table 5.4. List of possible encodings

Identity-H
Identity-V
Cp1250
Cp1252
Cp1257
MacRoman

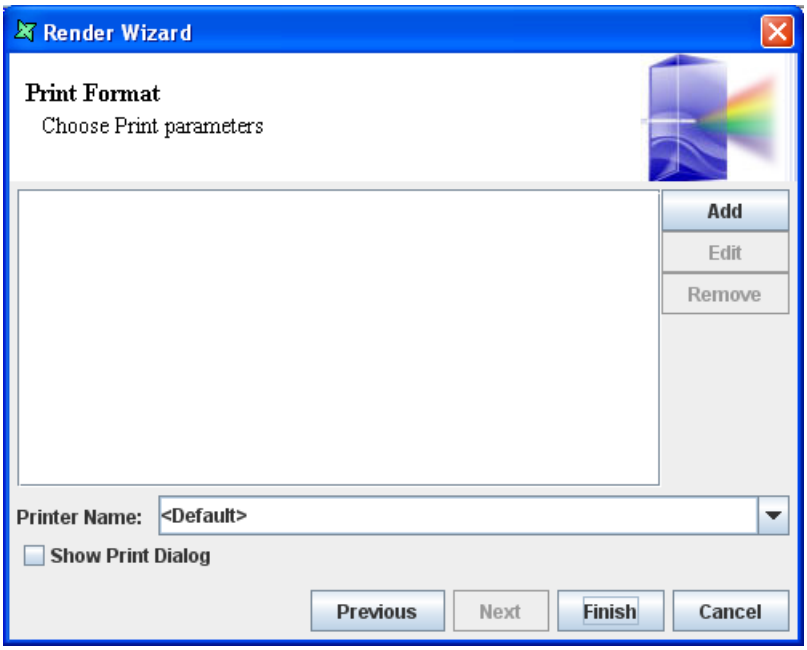
Note

Many fonts are released under a license that prevents embedding. You must ensure that you are allowed to embed the fonts you use.

Print

This format is selected if you want to print the reports directly. If you select Print as your output type the Launch to Viewer and the Save to Disk check boxes are disabled. Click the Next button. The screen appears as shown in Figure 5.7, “Print Options”.

Figure 5.7. Print Options



On clicking the Add button the Print Attribute dialog window appears. There are a number of Attribute Names to choose from:

Chromacity: When this attribute is selected in the combo box the Chromacity combo box appears from which the monochrome or color option can be selected. Of course, this won't let you print colour on a black and white printer!

1. Monochrome: When this option is selected the report pages are printed in shades of gray.

2. Color: The report pages are printed in color (subject to printer support, of course).

Copies: This attribute is used to specify the number of copies to be printed. On selecting this option the Copies field appears in which the number of copies are specified.

Job Name: This attribute is used to specify the name of the print job. A job's name is an arbitrary string defined by the client. This does not need to be unique between different jobs.

NumberUp: This attribute is used to specify the number of print stream pages to impose upon a single side of an instance of the selected medium. That is if the number up value is 4, the printer must place 4 print stream pages on a single side of the paper.

Page Range: When this option is selected, you can control the range of print stream pages that the printer object uses for each copy of the document.

Printer Resolution: This attribute is used to specify the exact resolution supported by the printer or to be used for a print job.

Print Quality: This attribute is used to specify the print quality that the printer uses for the job. There are three Print Quality options Draft, Normal or High. Draft is the lowest quality available on the printer.

Sides: This attribute is used to specify how print-stream pages are to be imposed upon the sides of an instance of selected medium. There are various options such as Duplex, tumble, etc.

Note

For a more detailed information of the attribute types and their uses refer to the javax.print.attribute.standard package under the link http://java.sun.com/j2se/1.4.2/docs/api

PS

Postscript is a programming language optimized for printing graphics and text (whether on paper, film, or screen). This language was invented by Adobe Inc for communicating with the printers. It is an object oriented language meaning that it treats images, including fonts as collections of geometrical objects rather than as bitmaps.

There are currently no render properties for Postscript.

PPT

PPT renderer is used to generate a ppt file. The generated output is compatible with PowerPoint 2007 or later. Users using PowerPoint 2003 edition can view the rendered output provided they had installed the Microsoft Office Compatibility Pack.

This function allows the contents to be rendered into ppt slides.

Note

Report are rendered in the latest version of PowerPoint. The file can be read and viewed by older is installed.

RTF

The Rich Text Format(RTF) specification describes a method of encoding formatted text and graphics for easy transfer between applications, especially word-processors.

The RTF render properties allow three pieces of information to be stored in the output file: the Author, the Company and any Comments. The method of processing images appears to have been corrected in Microsoft Word 2003 - they were scaled wrongly before. So if you want to make the RTF file

Microsoft Word 2003 compatible then the check box has to be turned on. If this option is switched off then Elixir Report Designer generates an RTF file that includes a workaround to make it compatible with older versions of Microsoft Word, but which will make images too large in Microsoft Word 2003.

For RTF elements with coloured backgrounds, OpenOffice 2.x will not be able to display the background colour as it is not supported.

SVG-Zip

The Scalable Vector Graphics (SVG) specification is a W3C standard for describing vector (and bitmap) graphics. Browsers may support SVG through native code (for example Mozilla Firefox) or through a plugin.

This renderer outputs in Paged mode and provides a separate svg file for each page of the report. The set of svg files is zipped along with a set of HTML navigation controls that allow interaction with the set of pages.

XLS

The XLS Renderer is used to generate an Excel file. The generated output is compatible with Excel 97 or later.

XLS provides a Snap option that may improve (or worsen) the layout of Excel cells. When Snap is off, the cells are positioned exactly the same as in the report template. This means if there is a one pixel gap between two report elements, there will be a one pixel row or column in the Excel file. When Snap is enabled, all coordinates are adjusted to fit on a 4x4 pixel grid. This means a component at 9,11 will be adjusted to 8,8. This adjustment removes the very small cells used as spacers, but may adjust the layout slightly to accomodate the grid. You should try both Snap on and Snap off to see which works best for your template. In general, if you use HBox and VBox to do alignment, then Snap off will usually give the best results.

XLS provides a Force numbers as text option which allows the full format of the numbers to be displayed. When Force numbers as text option is off, the numbers rendered in excel output will be displayed in a whole figure format. When the Force numbers as text option is enabled, the numbers rendered in excel output will be displayed in a complete format including the decimals.

XLS supports Mupltipler property. XLS allows the setting of cells width scale of excel output. To configure the cell width scale, enter a appropriate value in Multipler textfield. To allow lengthy words to be displayed fully, larger multiplier value has to be entered.

XLS supports the Paged render property. When the Paged check box is not selected, the data will be displayed on one continuous worksheet in the Excel file. If the Paged check box is selected, the data will be split into multiple pages (based on the page setup) and displayed across multiple worksheets. When Paged is false, no page headers or footer will be rendered.

XLS will take the `Left Margin` and `Top Margin` settings done in `Page Setup` and apply it when rendering.

Note

There is a limit to the number of rows that spreadsheets can display in one sheet. The limit varies between products and versions, but is around 32,000 rows. Paged option is not applicable to `On Render Begin`.

XML

Elixir Report Designer is able to export the report as an XML file. The XML render properties allow the setting of the XML encoding type and whether the output is Indented. As with other formats, you can choose paged or streamed mode of output. In paged mode the output will be broken into logical pages with headers and footers according to the page setup. In streamed mode the report will render a single flow of elements, without any page headers or footers. From Repertoire 7 onwards, XSLT is supported.

Availability of Renderer

EREngine-config.xml located at Elixir Repertoire\config controls the availabilty of renderer such as xls, ppt etc as listed above.

For example,

```
<!--render name="application/vnd.ms-excel" class="com.elixirtech.
  report2.engine.XLSPipeline"/-->
```

indicates the xls renderer has been disabled. In other words, the xls renderer had been mask off. To enable the renderer, simply remove the comment, as shown below.

```
<render name="application/vnd.ms-excel" class="com.elixirtech.
  report2.engine.XLSPipeline"/>
```

Chapter 6
Scripting with JavaScript

Overview

Elixir Report Designer supports JavaScript 1.5 as standardized by the European Computer Manufacturer's Association(ECMA) as ECMA Script version 3. This document details some of the features of JavaScript and their use within the Report framework.

In Elixir Report Designer, scripts are always rendered in the order that the bands appear in the output. Within a band, scripts in `RenderIf` will be rendered first. It will be followed by `OnRenderBegin`, `OnRenderEnd`, then `OnLayout`. As for the element(s) in each band, scripts of each element will run in the order the elements appear in the `Shapes` tree (Z-order).

JavaScript is a weakly typed, object-based scripting language modeled on Java syntax. It is a case-sensitive language and the following key-words are reserved. It is a case sensitive language and the following keywords are reserved.

Table 6.1. JavaScript Keywords

break	else	instanceof	true
case	false	new	try
catch	finally	null	typeof
continue	for	return	var
default	function	switch	void
delete	if	this	while
do	in	throw	with

Three primitive types are available: Numbers, Booleans and Strings, along with two compound types: Objects and Arrays.

Number

Numbers in JavaScript are represented internally using 64-bit floating point values - there is no integer type. Numbers may be written in decimal form: 10, 20.5, 30, exponent form: 3e-2 or hexadecimal: 0xFF.

JavaScript supports all conventional mathematical operators, including +,-,*,/ and % (modulus). The Math object provides additional functionality, such as sqrt, power, sin, cos, tan etc. Certain operations may yield errors with invalid input, for example Math.sqrt(-1) produces the result NaN - a special value which indicates the result is Not A Number.

Boolean

The Boolean type has two values: true and false. Boolean can be used as a function: Boolean(x) where the result is true unless x is 0, NaN, null, undefined or "" (the empty string).

String

The String type represents a sequence of characters and is delimited with either single or double quotes: "Hello World", 'Elixir Report'. Strings can be concatenated with the + symbol:

```
"Hello " + "World"
```

Other types are automatically converted to Strings when concatenated with them:

```
count = 5; message = "Hello" + count
```

This example also shows that multiple statements are separated by semi-colons. Strings support many additional functions, including: length, charAt, indexOf, match, replace, split, substring etc. Strings in JavaScript are immutable - just like in Java - once created they cannot be changed.

Object

Objects are compound types because they can contain multiple properties (named values of any type). Elixir Report Designer provides objects which represent all of the components that can be manipulated through the Report Designer. Most use of scripts in Elixir Report Designer consists of manipulating the properties of these objects during the report rendering process.

Where JavaScript is used to access Java objects, it is possible to use two alternate forms of access. You can access properties through traditional get and set methods, or access the properties directly, which JavaScript then translates into the appropriate get and set operations. For example, these two lines have identical behaviour:

```
component.setBackgroundColor("Red");
component.backgroundColor = "Red";
```

A similar situation applies for get:

```
c = component.getBackgroundColor();
c = component.backgroundColor;
```

You can choose the style of property access that suits you.

Objects loaded from a datasource may sometimes have field names that are not valid names in JavaScript. Such fields can still be accessed by using a named lookup of the value. For example, if the field name is 2000 (which obviously isn't a valid variable name) you can access the value with

```
this["2000"]
```

Array

Arrays are compound types which contain indexed values of any type. An array is created using the new keyword:

```
a = new Array(10)
```

This allocates ten slots in the array named a, that are indexed a[0] - a[9]. Arrays can also be created with initial values using an alternate syntax:

```
a = ["Hello", "World", 123 ]
```

This creates an array named a with three slots, where a[0] is the string "Hello", a[1] is the string "World" and a[2] is the number 123.

References

For more detailed information on JavaScript, please refer to:

- JavaScript Pocket Reference, David Flanagan, (O'Reilly)
- JavaScript: The Definitive Guide, David Flanagan, (O'Reilly)
- ECMAScript Language Specification (ECMA-262) (www.ecma.ch)

Scriptlets

Elixir Report Designer allows small scripts to be executed to control or interact with the rendering process. Within each script, the keyword `this` identifies the current template component - which might be a ReportHeader, PageHeader, Field, Image etc. See the list of objects in the Elixir Report Designer JavaScript API. The `this` keyword is optional in the scripts:

```
this.setFontColor("Red")
```

is equivalent to:

```
setFontColor("Red")
```

Each part of a report - from the Report object itself, down to Sections, Report Bands (eg. a Group Header), and every component can have up to four scripts: RenderIf, OnRenderBegin, OnRenderEnd and OnLayout associated with it. The scripts will be executed based on the order the objects are rendered. For report bands, this is document order, for components it is the order they appear in the shape tree (Z-order, from back to front). OnRenderEnd and OnLayout will occur after all child components have been rendered, so Report OnRenderBegin will be the first script to run and Report OnRenderEnd the last.

```
RenderIf -- except for Report
OnRenderBegin
-- any child components rendered here
OnRenderEnd
[OnLayout] -- only applies to report bands
```

RenderIf

RenderIf is evaluated to determine whether a component should be rendered. If the script evaluates to true, the component is rendered normally, if it evaluates to false, the component is not rendered and occupies no space in the output. Here's a sample renderIf script:

```
Parameters.get("user")=="Elixir"
```

This script returns true if the value of the dynamic parameter "user" is equal to the value "Elixir", otherwise it returns false. Thus the component will only be rendered if the "user" is "Elixir".

If no renderIf script is defined the default is true - the component is rendered.

OnRenderBegin

OnRenderBegin is executed before each component is rendered. For example, if the report has ten details, onRenderBegin will be called on each component in the detail section ten times, once per render of the component. The keyword `this` identifies the component being renderered. Here's a sample onRenderBegin script:

```
if (count%2==1)
{
    setBackgroundColor("Yellow");
}
```

This script checks the count variable. If the number is odd (1,3,5 etc,) the background colour of the component is changed to "yellow". Note that this modified template component will be used for all further renderings, so all subsequent even renderings (2,4,6 etc.) will be yellow as well - that's probably not what we intended, so it would often be advisable to include:

```
else
{
    setBackgroundColor("White");
}
```

to ensure changes to the template are not propogated to subsequent details.

OnRenderEnd

OnRenderEnd is executed after each component is rendered. This script is useful for incrementing counters and modifying the results of rendering (eg. formatting) before the report is finalized. The `this` keyword still refers to the template component used to render the output. Another variable `result` now references the rendered component. Here's a sample onRenderEnd script:

```
field = result.getLogicalElement(0);
if (field.getText()<0) result.setBackgroundColor("Red");
```

This script extracts the text from a field, checks whether the value is less than zero. If it is, the background colour gets set to red. Notice JavaScript will automatically turn the field text - a string - into a number for comparison. In this case, there's no need to provide an alternate colour for a positive result as the change is to the result element, not the template element. Changes to `this` affect the rest of the rendering from this template, whereas changes to `result` only affect the generated report.

Here's another:

```
field = result.getLogicalElement(0);
field.setText(field.getText().toUpperCase());
```

This script extracts the text from a field component and changes the formatted text to upper case. This can't be done in onRenderBegin, because the formatting is only performed during rendering. The call to getLogicalElement is required to conform to the logical RML structure, which is explained in detail in the JavaScript API section below.

OnLayout

OnLayout is only applicable for bands in the report. When a particular band, for example a Detail, has been rendered the OnRenderEnd script is called. Following this, the band may be paginated - even split into multiple chunks so that it can fit within the page boundaries. As this task is done, immediately after each chunk has been added to a page, the OnLayout script is invoked. At this point you can access `this`, `result` (the same as OnRenderEnd) and `page` - the page object on which this chunk has been put.

OnLayout is useful for controlling page-based information - for example, a page total that is displayed in a page footer. It is only at the point of pagination, after OnRenderEnd, that we know what page (or pages) a particular band will be assigned to. Hence this is the time to update page totals. The OnLayout for a chunk will be invoked before the OnRenderBegin of the Page Footer belonging to the Page that it is placed on. Here is an example sequence of scripts when the Detail band fits at the bottom of the page:

1. Detail RenderIf // assume returns true
2. Detail OnRenderBegin
3. Detail OnRenderEnd
4. Detail OnLayout // because it fits the page
5. PageFooter RenderIf // assume returns true
6. PageFooter OnRenderBegin
7. PageFooter OnRenderEnd

Here is the same sequence when the Detail band doesn't fit on the page:

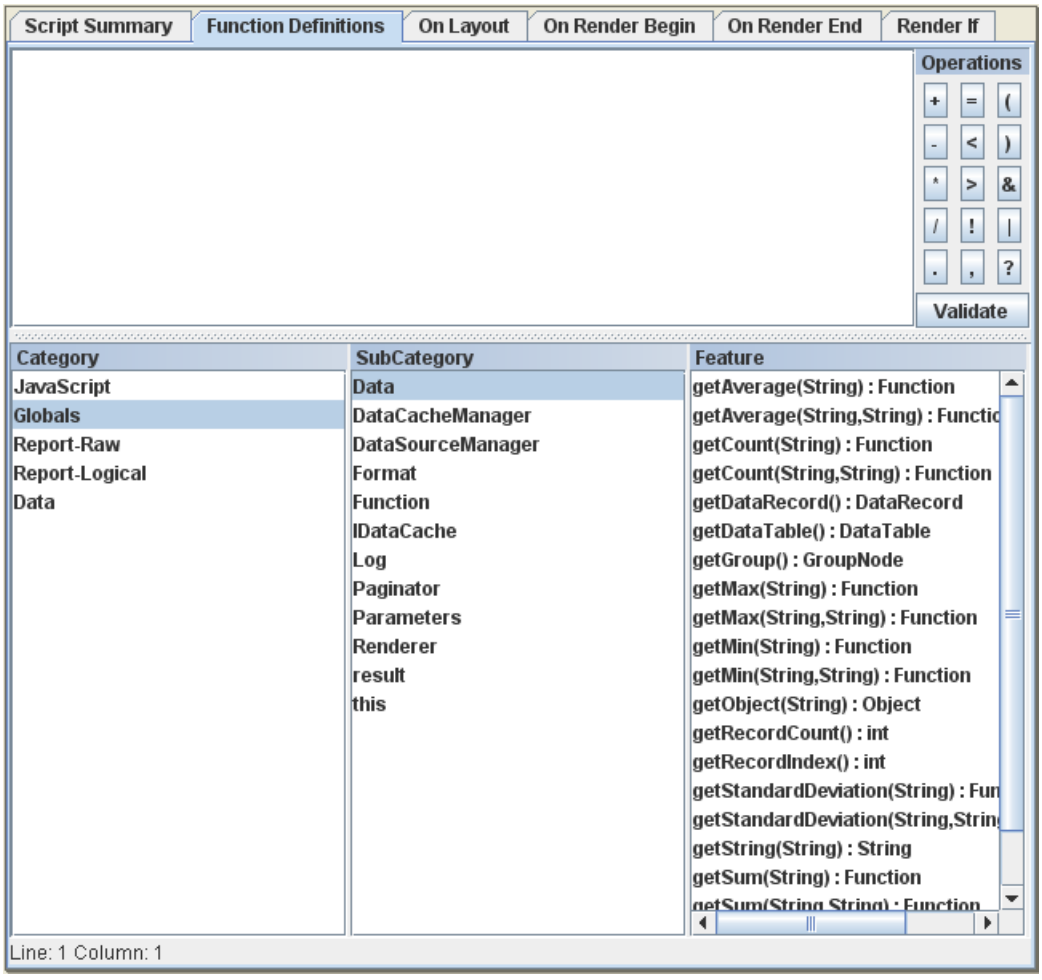
1. Detail RenderIf // assume returns true
2. Detail OnRenderBegin
3. Detail OnRenderEnd // by the end of this we know how big the detail is, and can see it won't fit
4. PageFooter RenderIf // assume returns true
5. PageFooter OnRenderBegin
6. PageFooter OnRenderEnd
7. PageHeader RenderIf // next page now - assume returns true
8. PageHeader OnRenderBegin
9. PageHeader OnRenderEnd
10. Detail OnLayout // delayed to here because the detail didn't fit before

Script Editor

To assist in editing scripts, the Functions panel includes a Script Editor as shown in Figure 6.1, “Script Editor” that provides access to the complete set of JavaScript objects and functions that are installed in the Elixir Repertoire release. This is a comprehensive set of APIs - you probably only need to refer to 5% of them. Many are for advanced JavaScripting, which is beyond the scope of this manual.

The Script Editor shows Categories, SubCategories (which are often Objects) and Functions in lists across the bottom. Choose an item from each list, starting from the left to explore the available functions. Once you have located a function or Object that you want to use, you can double-click on it to add it to the editor at the current cursor location. The cursor will then advance to the next logical place for text entry.

Figure 6.1. Script Editor



Another useful feature is the `Validate` button on the right. This allows the JavaScript syntax in the editor to be validated. As JavaScript is a weakly typed language, the check is for syntax only, not semantics. For example, entering `Data.getCount()` won't remind you that getCount expects a parameter, but entering `Data..getCount("Name")` will warn you that you have two '.' characters - this is a syntax error, whereas spellings and parameters are semantics. When validate is successful, a tick will appear next to the button, which will now say "Valid". Any text editing will return the button to it's original "Validate" state, allowing you to check again. Upon a validation failure, you will be given a message from the JavaScript compiler and the cursor will be placed at the point the error was detected.

JavaScript Security

Introduction

Several components of Elixir Repertoire such as data source and report template may embed Javascript to generate dynamic effects during report rendering.

Javascript is a powerful tool and may perform dangerous actions that will cause undesirable effects if not used properly. As a result, a user may want to control what the scripts can do. For example, javascripts should not be allowed to write any file on the system or establish any network connection to an unknown network.

Elixir Runtime enables to user to set the exact security permissions to the Javascript. This is based on the standard Java Security Architecture. Java 2 platform supports a policy-based, easily configurable and fine-grained access control model. Following this model, security permissions for Javascripts can be specified in a policy file, which is enforced by Java at runtime.

Steps to protect users from malicious Javascripts

- Find out the requirements of the application's security
- Configure security permissions given to application, Repertoire Runtime and Javascripts embedded in report templates
- Run the Java application with Security Manager
- Verify that security policies are taking effect

To find out the application's security requirements, you will need to check :

- The security permissions that should be given to the application. Once Security Manager is installed with the Java application, all Java classes are guarded by security policies, including your own codes. Therefore, if the application needs to perform security sensitive operations, you will need to grant those permissions explicitly.
- The security permissions that should be given to Repertoire's Runtime. Enough permissions should be given so that report templates can be loaded from the file system, write rendered reports to hard disk, etc. successfully.
- The security permissions that should be given to your Javascripts. This will depend on the functionality of the scripts. For example, if one of the scripts needs to write the output to a file, you might need to grant `write` permission on the file (to the script) explicitly.

Configure Security Permissions

The Java Security Manager checks a file for security permissions. By default, the file is called "java.policy" under your Java runtime installation folder (${JRE_HOME}/lib/security/java.policy). You can also save your policies in a different file, but you will need to tell Security Manager the location to find the file. The file must follow certain syntax so that Security Manager can work properly.

The policy file in the /config directory of Elixir Repertoire (java2.policy) grants all security permissions to Java classes loaded from /bin, /ext and /lib directory of Elixir Repertoire, Javascripts with code base "http://www.elixirtech.com/system/javascript". These classes are either Repertoire's built-in classes or put in by system administrators, which makes it safe to trust (by default).

In the default policy file, Javascripts written by end users are allowed to :

- read all Java system properties, such as "java.version"
- read files in Repertoire installation directory
- read, write and delete files in OS's temp directory

If the scripts try to do other security related actions (such as writing a system file import for your operating system), your scrips will fail to run and some error logs will be generated.

On Repertoire Report Designer UI, you can view console logs which are generated as you render your reports. Logs are organized into different categories and one of them is "JavascriptLog". When your scripts fail to run for some reason, you can always check this console log to see the reason of failing.

When your script fails because it does not have certain permissions to complete the task, the log entry should look something like : Error evaluating script: java.security.AccessControlException: access denied (<permission required>)

Verify that Security Policies are Taking Effect

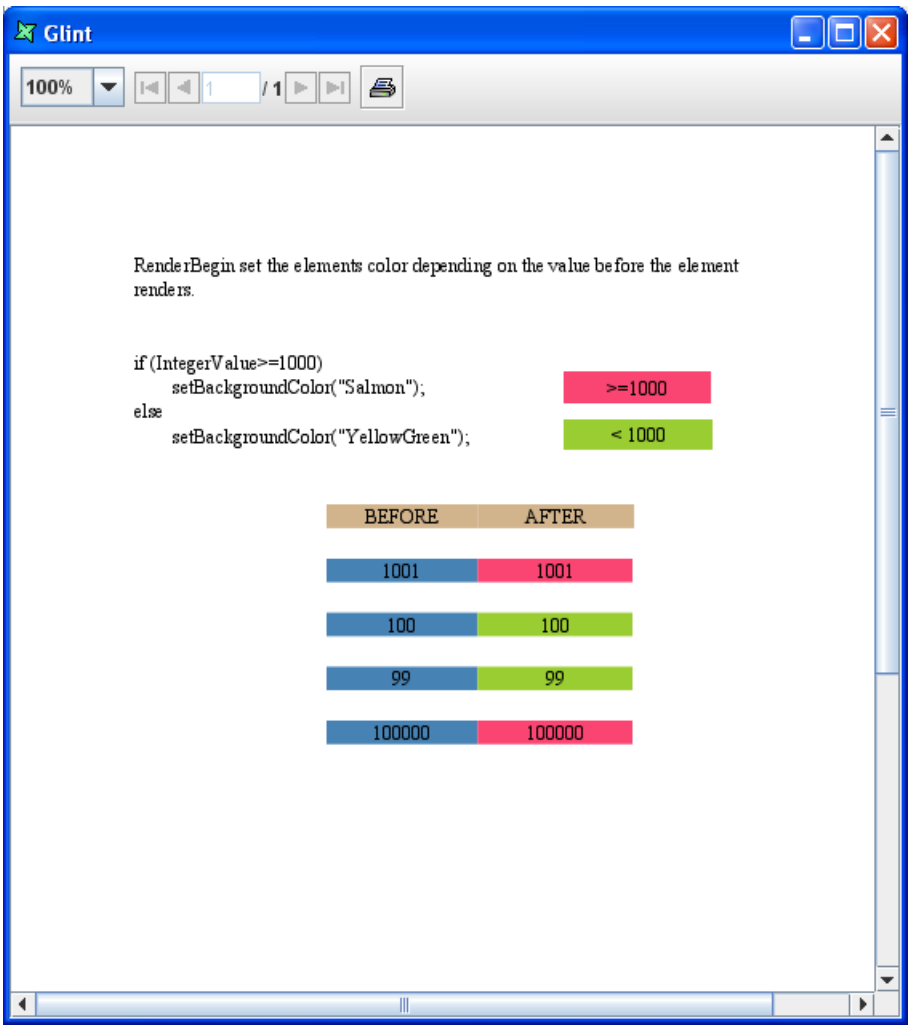
The best way to verify security policies is to write some Javascript. Try to do some operations that are not allowed and make sure the script fails.

Here is one short example :

When designing a report template with Repertoire Report Designer, you can insert Javascript in several places. There is one example given in ElixirSamples (under the /samples directory in Repertoire installation folder) which dynamically changes the background colour of some text field depending on the value being displayed. The report template used is /ElixirSamples/Report/Scripting/OnRenderBegin.rml.

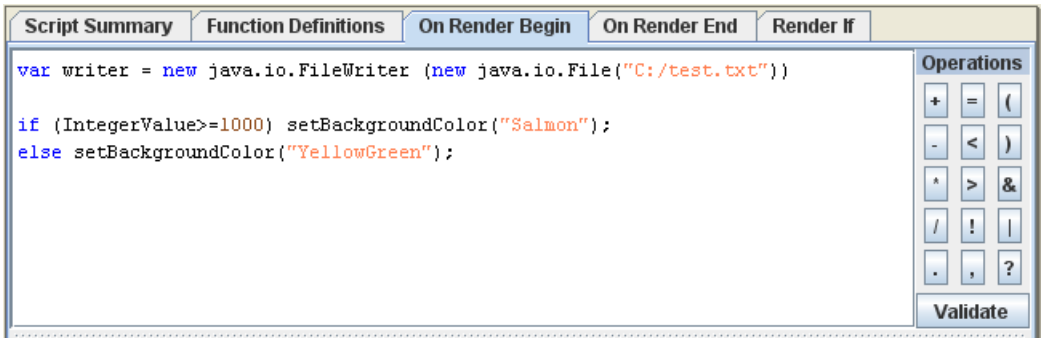
When you try rendering this template in Glint (without making any changes), the following figure, Figure 6.2, “Generated Report Before Changes”, will be the output.

Figure 6.2. Generated Report Before Changes



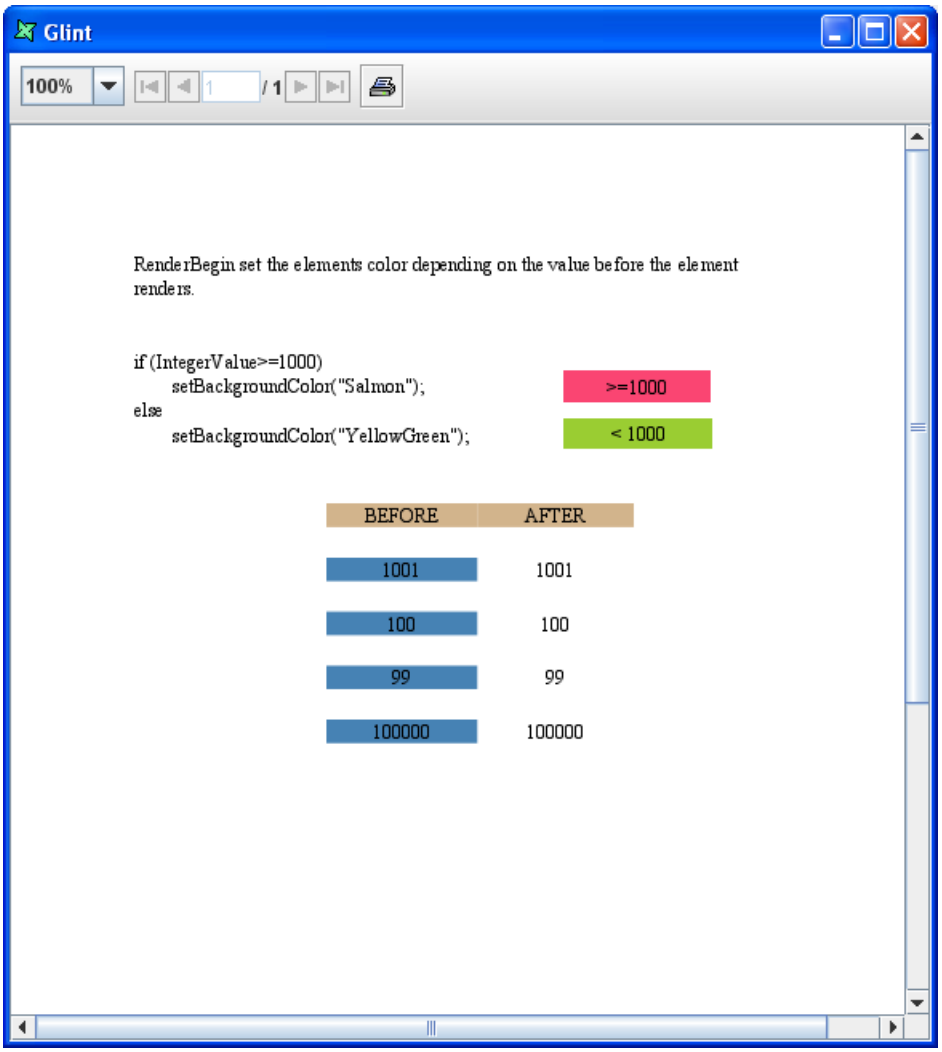
When the first line of script (as seen in Figure 6.3, “Additional Scripts”) is added to the report, the template generated will be different from the one shown above.

Figure 6.3. Additional Scripts



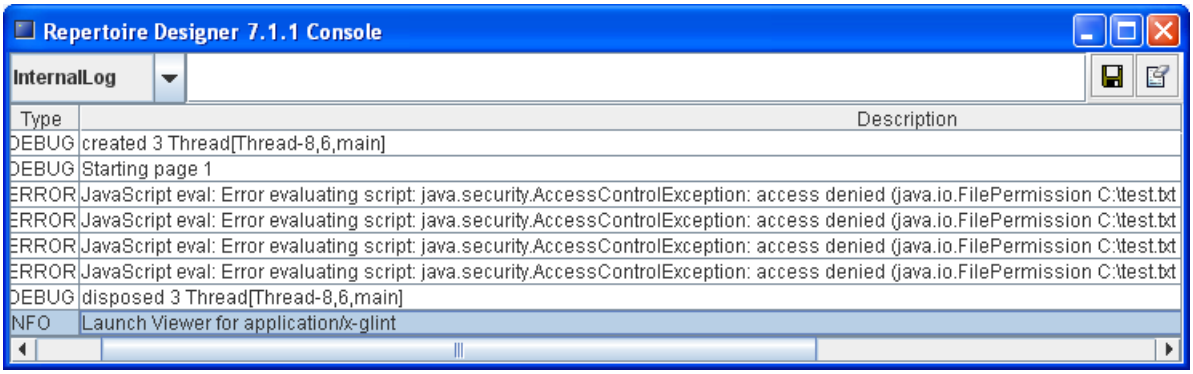
The generated output will be as seen in Figure 6.4, “Generated Report After Changes”.

Figure 6.4. Generated Report After Changes



In the console, error logs as seen in Figure 6.5, “Error logs in Console” will be captured indicating that the file is unable to write to the system.

Figure 6.5. Error logs in Console


Repertoire Designer 7.1.1 Console
InternalLog
Type
Description
DEBUG created 3 Thread[Thread-8,6,main]
DEBUG Starting page 1
ERROR JavaScript eval: Error evaluating script: java.security.AccessControlException: access denied (java.io.FilePermission C:\test.txt
ERROR JavaScript eval: Error evaluating script: java.security.AccessControlException: access denied (java.io.FilePermission C:\test.txt
ERROR JavaScript eval: Error evaluating script: java.security.AccessControlException: access denied (java.io.FilePermission C:\test.txt
ERROR JavaScript eval: Error evaluating script: java.security.AccessControlException: access denied (java.io.FilePermission C:\test.txt
DEBUG disposed 3 Thread[Thread-8,6,main]
NFO Launch Viewer for application/x-glint

This is an expected behavior as the settings in our default policy file (java2.policy) do not allow user scrips to write a file to the system and the script is trying to write a file named "test" into the system.

JavaScript Cookbook

Alternating colours

To alternate colours in a series of details, declare a count variable in the report header or the group header, if you are grouping.

```
// header onRenderBegin
count = 0
```

In the detail, set the colour based on the modulus of the count and increment count:

```
// component onRenderBegin
if (count%2==0) setBackgroundColor("Yellow")
else setBackgroundColour("White");
++count;
```

Hiding and showing components

If you have several components to hide based on a value, declare a boolean that you can check in renderIf:

```
// header onRenderBegin
today = new Date();
weekend = today.getDay()==0 || today.getDay()==6;
```

Now you can check for a weekend (Sunday==0, Saturday==6) and render accordingly:

```
// component renderIf
weekend
```

or for weekdays, use not weekend:

```
// component renderIf
!weekend
```

Using parameters to dynamically set values

Dynamic parameters can be passed in to a report at the start of rendering. These can be queried using the Parameters object. For example, to indicate whether a report is a Customer copy or a Sales copy you could specify the label to read a dynamic parameter called "type":

```
// label onRenderBegin
setText(Parameters.get("type"));
```

Other parts of the code could show or hide components based on the value of type:

```
// component renderIf
Parameters.get("type")=="Sales"
```

Accessing Java classes

Standard Java classes can be accessed by JavaScript. For example:

```
java.lang.System.out.println("Hello World");
```

The import mechanism can be used to avoid typing lengthy package names:

```
importClass(java.util.Vector);
v = new Vector();
importPackage(java.lang);
System.out.print("Hello ");
System.out.println("World");
```

When importing packages and classes which are not in the standard java runtime, the prefix Packages is required:

```
importClass(Packages.org.apache.log4j.Logger);
importPackage(Packages.org.apache.log4j);
```

Typically imports should be done in the FunctionDefinitions script so that they are accessible throughout the report. Any class that is available on the Elixir Report Designer classpath can be loaded and used in this way.

Chapter 7
Elixir Report Designer JavaScript Reference

JavaScript API Reference

Each object is described in terms of functions and properties. The Property identifier is used to indicate the ability to both get and set a value. For example,

```
Property: A : String
```

indicates that the object has both String getA() and void setA(String a) functions. If the return type is a Boolean, then the naming follows the Java convention of using "is" in place of "get" - Boolean isA and void setA(Boolean a).

Where objects are identified as X extends Y, this indicates that all functions of Y are applicable to each X object.

Where colours are required to be provided as Strings, Elixir Report Designer supports all CSS standard names, including: "Black", "Blue", "Cyan", "Magenta", etc. See Figure 4.2, “Named Colours” for a complete list.

```
this.setBackgroundColor("Red");
```

Alternatively, an RGB value may be supplied, with an optional alpha transparency value. Values should be in the range 0-255:

```
this.setBackgroundColor("rgb(255,0,0)"); // red,green,blue
```

```
this.setBackgroundColor("rgb(255,0,0,128)"); // red,green,blue,alpha
```

Elixir Utility Functions

To avoid possible conflicts with other libraries in future, these functions which were previously global, have been moved to a namespace `elxfn`. For backwards compatibility they are still also provided as global functions, but these will eventually be withdrawn. A JavaScript warning will be logged for each use of a deprecated function.

Table 7.1. Core Object, Number and String functions

- Boolean elxfn.isNull(a)
- Boolean elxfn.isNumber(a)
- Boolean elxfn.isString(a)
- Boolean elxfn.isDateString(str,pattern)

  The pattern parameter is optional, the default is "yyyy-MM-dd" if it is not defined.
- Boolean elxfn.isNumberString(str)

  Indicates whether a given string contains a valid number (parsed by Java). An alternative pure JavaScript approach is to use "new Number(str)!=NaN".
- Number elxfn.asNumber(str)
- String elxfn.trim(str)
- String elxfn.strip(str, ch)
- String elxfn.leftTrim(str)
- String elxfn.rightTrim(str)
- String elxfn.terminate(code,message)

  The terminate function provides a scripted way to abort report rendering. For example, if you want the report to only render when there are data records available, you can add a script to the first Section Header OnRenderBegin:

```
if (Data.getRecordCount()==0)
    elxfn.terminate(101,"No records available");
```

  The status code (101 was chosen here) will be passed back to the caller though the JobInfo. Values less than 100 are reserved for internal use. You could also use terminate to test for valid combinations of parameters.

Table 7.2. Date functions

- Date elxfn.offsetDays(origDate, numDays)
- Date elxfn.offsetMonths(origDate, numMonths)
- Date elxfn.offsetYears(origDate, numYears)
- Date elxfn.offsetDate(origDate, numYears, numMonths, numDays)
- Number elxfn.dateDiff(DateA, DateB, interval)

The allowed interval values for dateDiff are: "s","m","h","d" (seconds,minutes,hours and days).

Utility Objects

Table 7.3. Format

The Format object provides functions for manipulating and formatting text. The pattern definitions are described in the Java API.

- String padLeftAligned(String s,int width)
- String padRightAligned(String s,int width)
- String padCenterAligned(String s,int width)
- String spaces(int len)
- String formatNumber(Number n, int decimalPlaces)
- String formatNumber(Number n, int decimalPlaces, String locale)
- String formatNumber(Number n, String pattern)
- String formatCurrency(Number n)
- String formatCurrency(Number n, String locale)
- String formatPercent(Number n, int decimalPlaces)
- String formatPercent(Number n, String locale, int decimalPlaces)
- String formatDate(Date d, String pattern)
- String formatDate(Date d, String pattern, String locale)

Table 7.4. Log

The Log object interfaces with the Log4J logging mechanism used throughout Elixir Report Designer.

- void error(String msg)
- void warn(String msg)
- void info(String msg)
- void debug(String msg)

Table 7.5. Properties

The Parameters object provides a helper function to access report parameters. If this function is called from within a subreport it will return the subreport value for the property, but if not found there will return the parent report value instead. To set the parameter value you need to choose which report should hold the value and use the setParameter method of that RawReport (which could be either a master or a subreport). Similarly you can get a parameter from a report directly (ignoring master-subreport inheritance) by using the RawReport function getParameterValue.

Properties is deprecated and is kept for backward compatibility.

- String get(String name)

Table 7.6. Renderer

The Renderer object provides an interface to the render engine. There are two functions here, `eval` and `exec` for invoking the JavaScript engine. These are only needed if you construct JavaScript expressions within JavaScript itself, and want to run them. Use `eval` when you expect a return value, use `exec` when no return is needed.

- Object eval(String src, String lang, String code)
  - src: The name of the code being evaluated - displayed in error messages
  - lang: The script language used - must be "javascript" at present.
  - code: The code to be evaluated
  - return: The result of evaluating the script
- void exec(String src, String lang, String code)
  - src: The name of the code being executed - displayed in error messages
  - lang: The script language used - must be "javascript" at present.
  - code: The code to be executed
  - return: None
- LogicalElement getLogicalElementByName()
- LogicalReport getLogicalReport()
- String getMimeType()
- RawElement getRawElementByName()
- RawReport getRawReport()
- void pageBreak()
- void pageBreak(boolean resetPageCount)

Data Objects

Table 7.7. Data

The Data object represents the master datasource during section rendering. There are also helper methods for performing operations on named datasources. A number of the operations here return Function objects. The Function can then be applied to the desired record scope. See the Function object described below for more details.

- int getRecordIndex()
- int getRecordCount()
- Object getCount(String)
- String getString(String fieldName)
- Object getObject(String fieldName)
- Function getAverage(String fieldName)
- Function getMax(String fieldName)
- Function getMin(String fieldName)
- Function getStandardDeviation(String fieldName)
- Function getSum(String fieldName)
- Function getVariance(String fieldName)
- Function getAverage(String ds,String fieldName)
- Function getMax(String ds,String fieldName)
- Function getMin(String ds,String fieldName)
- Function getStandardDeviation(String ds,String fieldName)
- Function getSum(String ds,String fieldName)
- Function getVariance(String ds,String fieldName)

Table 7.8. DataCache

Each DataCache object holds a data table in memory along with an index to the current record in the table. A DataCache is created by the DataCacheManager or by filtering an existing DataCache.

- void reset()
- void reset(Properties props)
- int getRecordCount()
- int getRecordIndex()
- boolean hasNext()
- void next()
- int getColumnCount()
- String getColumnName(int index)
- int getColumnIndex(String fieldname)
- Object getObject(int rowIndex, String fieldname)
- Object getObject(int rowIndex, int columnIndex)
- void moveToRow(int rowIndex)
- Object getObject(String fieldname)
- String getString(String fieldname)
- int seekTo(String fieldname, Object value)
- DataCache filter(String fieldname, Object value)

Table 7.9. DataCacheManager

The DataCacheManager provides an interface to load or discard DataCaches from memory.

- DataCache loadCache(String dsnname, Properties props)
- DataCache loadCache(String dsnname, Object cacheName, Properties props)
- boolean hasCache(Object cacheName)
- DataCache getCache(Object cacheName)
- void putCache(Object newCacheName, DataCache cache)
- void removeCache(Object cacheName)

Table 7.10. Function

A Function is returned by the Data object to represent the different operations that can be performed on data - for example getSum(fieldName). Given the function, you can execute it by specifying the range of records you want to be included in the result. For example,

```
Data.getSum("Salary").getValueOverAll();
```

- Object getValueOverAll()
- Object getValueOverGroup()
- Object getRunningValueOverAll()
- Object getRunningValueOverGroup()

Table 7.11. GroupNode

A GroupNode represents a range of records - typically as a result of a grouping operation. GroupNodes form a tree structure with a single root which encompasses the whole datasource, which is then subdivided into child groups recursively, based on the section grouping configuration.

- boolean hasChildren()
- Iterator getChildIterator() // child GroupNodes
- int getStart() // the first index in the group
- int getStop() // the last index in the group
- GroupNode getParent() // null if topmost group

Raw Report Objects

The current raw object is accessed as `this` in the renderIf, onRenderBegin, onRenderEnd and onLayout scriptlets.

See the RML-RawModel.pdf in /docs for full details of the raw objects.

Logical Report Objects

Logical Report objects are only accessible in onRenderEnd and onLayout. They are identified using the `result` variable. Most raw elements are converted to more than one logical element. For example, a raw Field is rendered as a logical Rectangle which contains one or more logical Text objects - one for each line of output. Accessing the Text within the result is illustrated (in the onRenderEnd example above) like this:

```
field = result.getLogicalElement(0);
```

Where field is the first line of text in the result rectangle. You can query the number of child elements with:

```
count = result.getLogicalElementCount();
```

Wherever possible work with the raw objects, rather than the logical ones, as the dealing with logical objects requires an intimate knowledge of the rendering process and logical objects are more likely to change as new features are added.

See RML-LogicalModel.pdf in /docs for full details of the logical objects.

Chapter 8
Office Report Template

DocX

Although DocX is a new file format for Word from Microsoft Office 2007, users with Microsoft Office XP and Microsoft Office 2003 can also render a DocX file. To view a DocX file, install the File Compatibility Pack which can be downloaded from Microsoft Download Center, http://www.microsoft.com/downloads/. With OpenOffice, the Compatibility Pack is not required to view.

Getting Started

Creating a DocX file Repertoire Designer or Remote Designer is not supported. Therefore, a manual creation through Microsoft Office is required. After the creation of a DocX file, the repository will need to be refreshed in order to be shown under the filesystem.

By right-clicking on the DocX file and selecting `View`, the DocX file will open in Microsoft Office for editing or viewing.

Comments

There are markers in the document that indicate the locations of the different bands of the template. Only the start of a comment and the text associated with the comment is important.

Some comments have special meanings to them and they are listed in Table 8.1, “Comments”.

Table 8.1. Comments

Comment	Meaning
Section	Indicates the start of data processing. This can be defined by any name.
Header	Presents information about the particular data group being view. For instance, the group name and a brief description about the data contained in the group which appears at the beginning of a new group of record. It has the same concept as *Group Header* in Repertoire Designer. This can be defined by adding the following into the comment: `/Elx/Header`
Subreport	Contains data from another report. It has the same concept as a Subreport element in Repertoire Designer. This can be defined by adding the following: `/Elx/Subreport`
Detail	Contains the main body of the report's data. The detail section prints individual records and repeat until all records have been printed. It has the same concept as *Detail* in Repertoire Designer. This can be defined by adding the following into the comment: `/Elx/Detail`

Comment	Meaning
Footer	Contains totals or similar information that appears at the end of each group. It has the same concept as *Group Footer* in Repertoire Designer. This can be defined by adding: `/Elx/Footer`
End	Refers to the end of the *Section*. This can be defined by adding `/Elx/End`

DataSources

Define DataSource File

Defining datasource file is identical to the idea of adding a datasource in Repertoire Designer and Remote Designer. It can be defined by entering:\n /Elx/DataSource/<*SectionName*>/Name/<*dataSourceName*>.ds.

For example:

```
/Elx/Datasource/Section1/Name/FruitSales.ds
```

Define DataSource Parameter

This is similar to adding datasource parameter(s) in Repertoire Designer and Remote Designer. It is defined as: /Elx/DataSource/<*SectionName*>/Parameter/Name/<*ParameterValue*>.

For example:

```
/Elx/DataSource/Section1/Parameter/Name/${Company}
```

Group and Sort Data

Grouping and sorting of data in DocX has the same concept as Group and Sort in Repertoire Designer and Remote Designer. Grouping is done for categorization of data into various logical collections. Sorting is for arranging the data in ascending/descending order. They are defined by: /Elx/DataSource/<*SectionName*>/Group/<*dsFieldName*>/<*SortOrder*>.

An example will be:

```
/Elx/DataSource/Section1/Group/City/Ascending
```

Note

There are four different types of sorting available. They are `Ascending`, `Ascending-Lexicographic`, `Descending` and `Descending-Lexicographic`. `Ascending-Lexicographic` and `Descending-Lexicographic` are known as `Ascending(Simple)` and `Descending(Simple)` in Repertoire Designer respectively.

Parameter

A parameter is "declared" when user enters a string like *${parameterName}* within a DocX report. A parameter is "defined" when a value is supplied. A report may have multiple declarations and mixed with normal text. For example:

```
My name is ${first} ${last}.
```

There are 2 ways of defining parameters.

1. Define Parameter Individually - Defining each parameter one by one in the DocX report. (It can be defined as: /Elx/Parameter/<*ParameterName*>/Value)

   For example:

   ```
   /Elx/Parameter/Text/${User##MyName}
   ```

2. Load Properties File - All parameters declared are to be placed in a Properties file together with the desired elements defined. It can be defined as: /Elx/ParameterFile/<*Properties-FileName*>.properties

   The parameters should follow the following format: <*ParameterName*>=Value For example:

   ```
   Password=${Password#password#MyPassword}
   Name=${User##MyName}
   PreferredChoice=${MyChoice#choice(Apple,Orange,Pear)#Pear}
   Date=${Date#date#2009-01-01}
   ```

   The following, Figure 8.1, “Parameters in DocX Report”, is how a DocX report will look with the parameters declared and using a Properties file. When rendered, the outcome will be as seen in Figure 8.2, “Rendered DocX Report”. For more information about rendering a DocX report, refer to the section called “Render DocX Report” at the end of this chapter.

Figure 8.1. Parameters in DocX Report

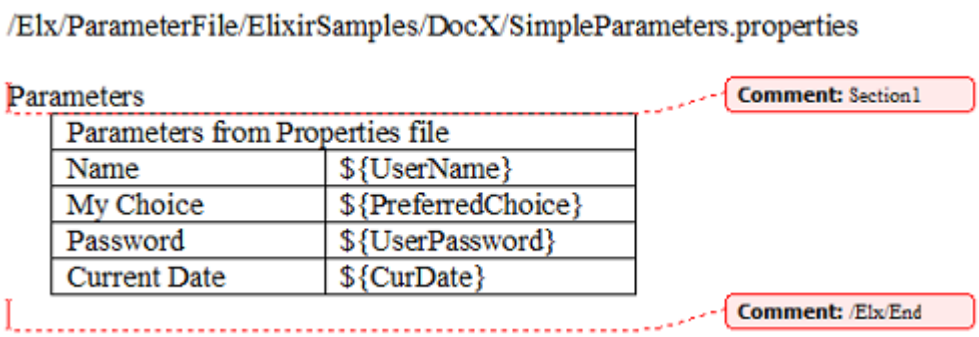


Figure 8.2. Rendered DocX Report

Parameters

Parameters from Properties file	
Name	MyName
My Choice	Pear
Password	MyPassword
Current Date	2009-01-01

Parameter Element

Each parameter can have up to three elements: name, type and default value. These elements are seperated by the character, #, which can be omitted when there are no subsequent elements. Here are some examples:

- ${name} - This syntax is used to specify a parameter name.
- ${name#type} - This syntax is used to specify the name and type of dynamic parameter.

- ${name#type#value} - This is used to specify the name, type and value.
- ${name##value} - This syntax is used to specify simply the name and value.

Scripts

Like the 2 ways of declaring parameters, scripts can either be declared individually or by loading a file, a script file.

1. Define Scripts Individually. (Defined as: /Elx/Script/<*Define scripts here*>)

   For example:

   ```
   /Elx/Script/function sumFn(a,b){return a+b;}
     var script2=sumFn(2,4);
   ```

   To use the script in the report, enter *${=script2}*.

2. Load a Script File. (Defined as: /Elx/ScriptFile/<*ScriptFileName*>.js)

   An example of the content in a .js file is:

   ```
   importScript("ElixirSamples/Reports/Scripting/minusJS.js");
   function sumFn(a,b)
     {
       return a+b;
     }
   ```

   To use the script in the DocX report, enter the following:

   ```
   ${=sumFn(2,4)}
   ```

Aliases

Aliases are used to shorten substitutions in order to fit the available space better. Take the following line for example:

```
${=elxfn.dateDiff(new java.util.Date(107,0,1),
  new java.util.Date(),'d')}
```

The result is probably a few integers which will fit in a table cell. However, Microsoft Word will try to show the whole substitution *${=elxfn.dateDiff(new java.util.Date(107,0,1),new java.util.Date(),'d')}* in the field, which will alter the layout. Therefore, for report neatness, we use substitution as an alias. (1 is used here, but any name is fine) Instead of a long string, we can use *${@1}* instead.

> **Note**
> The alteration of layout is only during the designing of DocX report. When rendered, the tables will display according to the height and length of the text.

Hide Processing Instructions

Processing instructions can be temporarily disabled by 2 ways.

1. Have an additional slash. For example:

   ```
   /Elx//Datasource/Section1/Name/FruitSales.ds
   ```

2. Use `Hide`. For example:

```
/Elx/Hide/Datasource/Section1/Name/FruitSales.ds
```

Formatting

Very often, substituted values need to be formatted. For example, a user wishes to show only the hours and minutes of a date, or show thousand seperator and two decimal places for calculation results. This is handled by prefixing the substitution with a format string, which always begins with the character, #. This follows the standard Java formatting syntax. The end of the string is marked with the | character.

Table 8.2. Formatting Syntax

	Syntax
Data Field	${%,.<*numOfDecimalPlaces*>f\|<*DataFieldName*>} An example will be: `${%,.2f|2000}`
Script	${%,.<*numOfDecimalPlaces*>f\|=<*JavascriptHere*>} For example: `${%,.2f|=var a=2;var b =4;totalSum=a+b;}`
Alias	/Elx/@/<*AliasName*>/%,.<*numOfDecimalPlaces*>f|=<*JavascriptHere*> For example: `/Elx/@/1/%,.2f|=var a=2;var b =4;totalSum=a+b;`

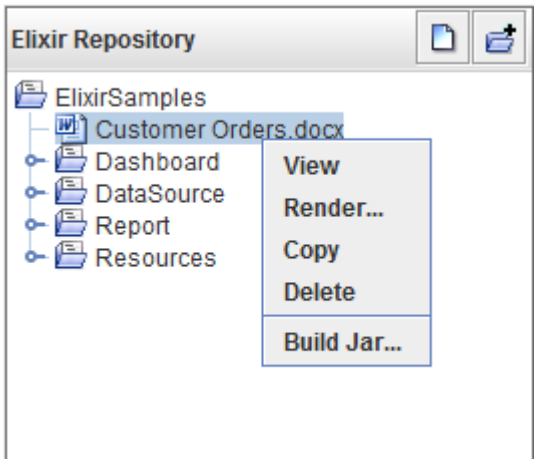
Note

As Javascript has no integer type, always use *%f* (floating point) when referring to numbers returned from Javascript. If you refer to *%d* (decimal integer), an error will occur. To show a floating number as an integer, use *%.0*. To show a numeric figure such as 12.34, use *%.2*, which indicates two decimal places.

Render DocX Report

For DocX files in the repository, on right-clicking on a particular DocX file as seen in Figure 8.3, “Render DocX” and select `Render...`. It can be rendered directly into DocX format with data from the datasource specified in the file.

Figure 8.3. Render DocX



Note

DocX rendering works in the Server, Repertoire Designer and Remote Designer across all compatible operating systems and not just Microsoft Windows. DocX rendering is not supported by the Report Runtime.

Chapter 9
Elixir Report Designer Migration Guide

Migration Guide

This document describes how to migrate from Elixir Report version 4 to Elixir Report version 5 or later. The document is intended for Elixir Report version 4 users and developers who are performing the migration.

The migrator for converting from version 4 to version 5 (or later) is included in all versions from Elixir Report Designer version 5 onwards. Although Elixir Report Designer is backwards compatible, the migration may not be 100 percent complete and templates may require some manual adjustment due to the changes listed below.

Migration Overview

Below is the list of report elements that will be handled by the migration process:

Template Migration

- Text component
- Label component
- Image component
- Line component
- Rectangle component
- Checkbox component
- Report Header/Footer
- Page Header/Footer
- Section Header/Footer
- Details
- Parameters
- Most Scheme Scripts
- All JavaScript Scripts

Data Source Migration

- Text Data Source

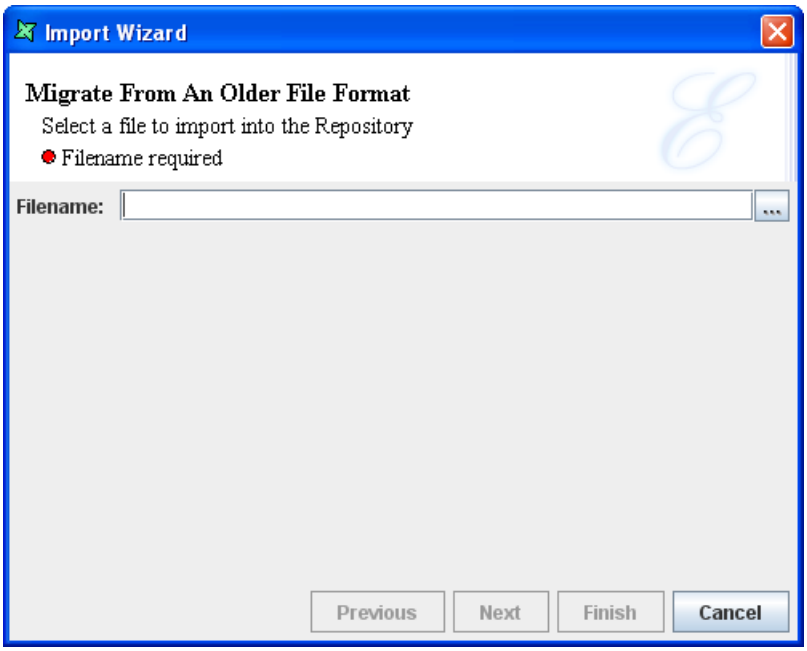
- JDBC Data Source
- XML Data Source
- XML2 Data Source

Template Migration Steps

1. Start Elixir Report Designer version 5 (or later). Click on the Toolbar icon at the top of the repository panel to create a new Local File system for report template.

2. Right-click at the Local File system that you have created. On selecting the `Import...`, the "Import Wizard" will appear as shown in Figure 9.1, “Import Wizard”. Enter the name of the Report Template (.template) in the text box. Alternatively, you can click on the `...` button to launch the File Chooser dialog to select the Report Template that you wish to import.

   Figure 9.1. Import Wizard



3. Click on the Finish button to complete the migration process. The name of the Report Template will be displayed under the Local File system in the repository panel to indicate that the migration is completed.

   ### Note

   This form of migration is file-by-file. See Batch Mode Template Migration below for an automated solution.

DataSource Migration Steps

1. Start Elixir Report Designer version 5 (or later). Click on the Toolbar icon at the top of the repository panel to create a new Local File system for report template.

2. Right-click at the Local File system that you have created. On selecting the `Import...`, the "Import Wizard" will appear as shown in Figure 9.1, “Import Wizard”. Enter the name of the

datasource file (.sav) in the text box. Alternatively, you can click on the `...` button to launch the File Chooser dialog to select the datasource file that you wish to import.

3. Click on the Finish button to complete the migration process. The names of the datasource will be displayed under the Local File system in the repository panel to indicate that the migration is completed.

Note

From version 5, each data source will be saved as separate .ds files in the chosen location instead of the single combined .sav file in version 4.

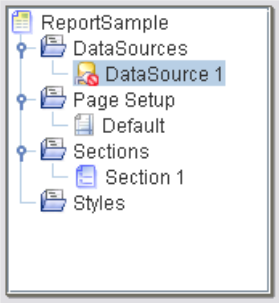
Adding a DataSource

After you have successfully migrated the templates and datasource, you will have to add the new data source to the new report template in order to render the report.

1. Double-click on the .rml file to launch the report template in the workspace.

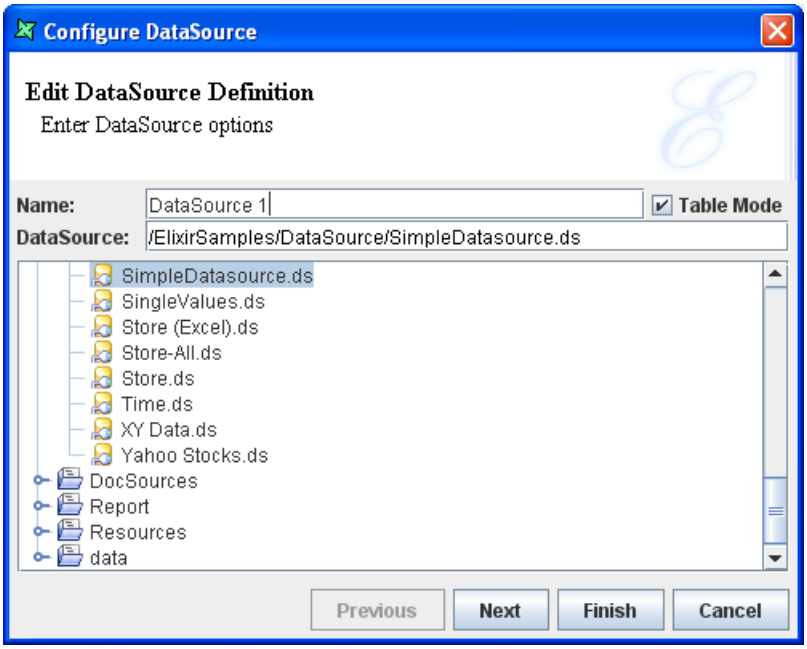
2. Under the Report tab, you will see a list of elements belonging to the current report template. An example is shown in Figure 9.2, “Report Elements”.

Figure 9.2. Report Elements



Right-click on the DataSource 1 node under the DataSources subtree and select `Edit DataSource...`. A Wizard will appear as shown in Figure 9.2, “Report Elements”.

Figure 9.3. Configure DataSource Wizard



3. In the "Configure DataSource Wizard", click on the appropriate .ds filename to select the datasource which you wish to use and click the Finish button to complete the process.

Batch Mode Template Migration

The Elixir Repertoire bin directory contains a migrate batch file and shell script that illustrate how to invoke the migrator module without choosing each template file explicitly using the GUI. You should always make a backup of your ER4 templates before attempting migration.

The main class launched for migration is:

```
com.elixirtech.report2.migration.ModelMigrator
```

and all the jars listed in the scripts must be on the classpath to use it. If you run the migrator with no parameters you get a help message as follows:

```
C:\ElixirRepertoire\bin> migrate
  Usage: ModelMigrator [-r] file...
  More than one filename can be specified, if a directory name
  is used all .template files within it will be processed. By
  using the -r option, directories will be processed recursively
```

So to convert all templates within a folder, just use

```
migrate C:\ER4\Samples
```

and all template files in that directory will be converted into the new RML format. The new files will be placed in the same directory, but with .rml extensions. To convert templates from multiple directories, use

```
migrate -r C:\ER4\Samples
```

and all templates in Samples and all sub directories will be converted. In all cases, the original templates will not be modified, however any existing rml files that have the same name as a migrated template will be overwritten without warning.

Note

The migrator used here in batch mode is identical to the one invoked from the GUI, therefore the same limitations on migration apply and some templates may still require manual adjustments.

Changes and Improvements in Version 5

There are many advances in features and functionality in version 5 and later. Please review the entire manual for details. This section focuses on areas of the rendering process that changed between version 4 and 5.

- There has been a change in the extension used for report templates and datasources to highlight the internal formats have been completely updated. Both file types still hold XML, but the new structure allows more features to be supported while reducing the file size. In version 4, a report template was saved with the `.template` extension, whereas in version 5 the extension is `.rml` (Report Markup Language). In version 4 the datasources were grouped together in a single `.sav` file, whereas from version 5 each datasource is stored as a separate `.ds` file in the repository. This makes for much faster loading and simplified deployment.

- In version 4, the rendering sequence was:

```
Report Header
----Page Header
--------Section Header
------------Details
--------Section Footer
----Report Footer
Page Footer
```

  From version 5, there can be multiple sections in a report, and each section has the following render sequence:

```
Page Header
----Section Header
--------Group Header
------------Details
--------Group Footer
----Section Footer
Page Footer
```

  As you can see, this is a more symmetrical arrangement. To avoid confusion, because version 5 onwards supports multiple Sections, the version 4 Section Headers are now called Group Headers, and Section Header/Footer refer to the header and footer of each entire report section. (If there is only a single section in the report, then these are effectively report header and report footer respectively.)

- From Elixir Report version 5 onwards, we have consolidated the scripting language for our products to be JavaScript which more convenient for non-programmers. The scheme functions provided in version 4 will be obsoleted in this version. Many scheme functions will be automatically converted into JavaScript during the migration, however in a few cases, especially with data caches, there is no corresponding API in version 5. This is because from version 5 Elixir tools are based on a true multi-threaded system, and the version 4 API made a number of assumptions that are not sustainable in a multi-threaded environment.

Data Sources

- XML DataSource: Elixir Report version 5 (or later) supports the XML2 datasource using XPath to query the XML document, locate and retrieve data.
- Enterprise Java Bean (EJB) Datasource: The EJB datasource is available through the Object Datasource from version 5.
- CrossTab Datasource: From Elixir Report version 5.0, the Crosstab features are available through the Composite DataSource.

Report Elements

Below is the list of report elements that will not be migrated automatically. Manual creation of the report element is required:

- Barcode Component
- Chart Component: This report element is replaced with charts rendered using JFreeChart from version 5.
- SVG Component
- Boxed-In-Text Component: This report element is replaced by the Data Grid component from version 5.
- HTML Component & Rich Text Format (RTF) Component: These report elements are supported through Callback component from version 5.
- Multi-Row Data Cache Viewer Component: This report element is replaced by the Data Grid component from version 5.
- Table Cache View Component: This report element is replaced by the Table component from version 5.

Special Parameters

Special parameters that were understood by the Elixir Report 4 engine are often handled differently in Elixir Report 5 and above. This means that parameters like "elixir.report.ds.parameter.keyset" will be migrated to ER5 parameters, but the engine will no longer utilise them. Such special parameters are no longer required in version 5 - all features can be set through the tool user interface, so there is no need to remember these long parameter names.

Render Formats

- XHTML Files rendered with Elixir Report 3.x method will be obsoleted from version 5.
- Java Graphics Format (JGF) export format is replaced by Glint vector graphics format which is more precise and compact as well as supporting additional features.

Runtime API

Elixir Report version 5 defines a new Runtime and Server API. Customers or developers migrating to version 5 or later will have to reengineer applications to work with the APIs. Runtime API Documentation is provided in the directory `runtime/api`.